IBM Spectrum Scale
Version 5.0.0

Administration Guide

IBM

IBM Spectrum Scale
Version 5.0.0

Administration Guide

IBM

This edition applies to version 5 release 0 modification 0 of the following products, and to all subsequent releases and modifications until otherwise indicated in new editions:
- IBM Spectrum Scale ordered through Passport Advantage® (product number 5725-Q01)
- IBM Spectrum Scale ordered through AAS/eConfig (product number 5641-GPF)
- IBM Spectrum Scale for Linux on Z (product number 5725-S28)
- IBM Spectrum Scale for IBM ESS (product number 5765-ESS)

Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

IBM welcomes your comments; see the topic "How to send your comments" on page xxvii. 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

Tables

About this information

This edition applies to IBM Spectrum Scale™ version 5.0.0 for AIX®, Linux, and Windows.

IBM Spectrum Scale is a file management infrastructure, based on IBM® General Parallel File System (GPFS™) technology, which provides unmatched performance and reliability with scalable access to critical file data.

To find out which version of IBM Spectrum Scale is running on a particular AIX node, enter:
```
lslpp -l gpfs\*
```

To find out which version of IBM Spectrum Scale is running on a particular Linux node, enter:
```
rpm -qa | grep gpfs       (for SLES and Red Hat Enterprise Linux)
dpkg -l | grep gpfs       (for Ubuntu Linux)
```

To find out which version of IBM Spectrum Scale is running on a particular Windows node, open **Programs and Features** in the control panel. The IBM Spectrum Scale installed program name includes the version number.

Which IBM Spectrum Scale information unit provides the information you need?

The IBM Spectrum Scale library consists of the information units listed in Table 1 on page xiv.

To use these information units effectively, you must be familiar with IBM Spectrum Scale and the AIX, Linux, or Windows operating system, or all of them, depending on which operating systems are in use at your installation. Where necessary, these information units provide some background information relating to AIX, Linux, or Windows. However, more commonly they refer to the appropriate operating system documentation.

Note: Throughout this documentation, the term "Linux" refers to all supported distributions of Linux, unless otherwise specified.

Table 1. IBM Spectrum Scale library information units

Information unit	Type of information	Intended users
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	This guide provides the following information: **Product overview** • Overview of IBM Spectrum Scale • GPFS architecture • Protocols support overview: Integration of protocol access methods with GPFS • Active File Management • AFM-based Asynchronous Disaster Recovery (AFM DR) • Data protection and disaster recovery in IBM Spectrum Scale • Introduction to IBM Spectrum Scale GUI • IBM Spectrum Scale management API • Introduction to Cloud services • Introduction to file audit logging • IBM Spectrum Scale in an OpenStack cloud deployment • IBM Spectrum Scale product editions • IBM Spectrum Scale license designation • Capacity based licensing • IBM Spectrum Storage™ Suite **Planning** • Planning for GPFS • Planning for protocols • Planning for Cloud services • Firewall recommendations • Considerations for GPFS applications	System administrators, analysts, installers, planners, and programmers of IBM Spectrum Scale clusters who are very experienced with the operating systems on which each IBM Spectrum Scale cluster is based

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	**Installing** • Steps for establishing and starting your IBM Spectrum Scale cluster • Installing IBM Spectrum Scale on Linux nodes and deploying protocols • Installing IBM Spectrum Scale on AIX nodes • Installing IBM Spectrum Scale on Windows nodes • Installing Cloud services on IBM Spectrum Scale nodes • Setting up a Cloud services cluster • Installing and configuring IBM Spectrum Scale management API • Installing Active File Management • Installing and upgrading AFM-based Disaster Recovery • Installing call home • Installing file audit logging • Steps to permanently uninstall GPFS and/or Protocols	System administrators, analysts, installers, planners, and programmers of IBM Spectrum Scale clusters who are very experienced with the operating systems on which each IBM Spectrum Scale cluster is based

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	**Upgrading** • IBM Spectrum Scale supported upgrade paths • Upgrading to IBM Spectrum Scale 5.0.x from IBM Spectrum Scale 4.2.y • Upgrading to IBM Spectrum Scale 4.2.y from IBM Spectrum Scale 4.1.x • Upgrading to IBM Spectrum Scale 4.1.1.x from GPFS V4.1.0.x • Upgrading from GPFS 3.5 • Online upgrade support for protocols and performance monitoring • Upgrading object packages • Upgrading NFS packages • Upgrading SMB packages • Manually upgrading pmswift • Manually upgrading the performance monitoring tool • Manually upgrading the IBM Spectrum Scale management GUI • Upgrading Cloud services • Upgrading to IBM Cloud Object Storage software level 3.7.2 and above • Upgrading IBM Spectrum Scale components with the installation toolkit • Migrating from Express Edition to Standard Edition • Completing the upgrade to a new level of IBM Spectrum Scale • Reverting to the previous level of IBM Spectrum Scale • Coexistence considerations • Compatibility considerations • Considerations for IBM Spectrum Protect™ for Space Management • GUI user role considerations • Applying maintenance to your GPFS system	System administrators, analysts, installers, planners, and programmers of IBM Spectrum Scale clusters who are very experienced with the operating systems on which each IBM Spectrum Scale cluster is based

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Administration Guide	This guide provides the following information: **Configuring** • Configuring the GPFS cluster • Configuring the CES and protocol configuration • Configuring and tuning your system for GPFS • Parameters for performance tuning and optimization • Ensuring high availability of the GUI service • Configuring and tuning your system for Cloud services • Configuring file audit logging • Configuring Active File Management • Configuring AFM-based DR • Tuning for Kernel NFS backend on AFM and AFM DR **Administering** • Performing GPFS administration tasks • Verifying network operation with the mmnetverify command • Managing file systems • File system format changes between versions of IBM Spectrum Scale • Managing disks • Managing protocol services • Managing protocol user authentication • Managing protocol data exports • Managing object storage • Managing GPFS quotas • Managing GUI users • Managing GPFS access control lists • Considerations for GPFS applications • Accessing a remote GPFS file system	System administrators or programmers of IBM Spectrum Scale systems

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Administration Guide	• Information lifecycle management for IBM Spectrum Scale • Creating and maintaining snapshots of file systems • Creating and managing file clones • Scale Out Backup and Restore (SOBAR) • Data Mirroring and Replication • Implementing a clustered NFS environment on Linux • Implementing Cluster Export Services • Identity management on Windows • Protocols cluster disaster recovery • File Placement Optimizer • Encryption • Managing certificates to secure communications between GUI web server and web browsers • Securing protocol data • Cloud services: Transparent cloud tiering and Cloud data sharing • Managing file audit logging • Highly-available write cache (HAWC) • Local read-only cache • Miscellaneous advanced administration • GUI limitations	System administrators or programmers of IBM Spectrum Scale systems

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Problem Determination Guide	This guide provides the following information: **Monitoring** • Performance monitoring • Monitoring system health through the IBM Spectrum Scale GUI • Monitoring system health by using the mmhealth command • Monitoring events through callbacks • Monitoring capacity through GUI • Monitoring AFM and AFM DR • GPFS SNMP support • Monitoring the IBM Spectrum Scale system by using call home • Monitoring remote cluster through GUI • Monitoring file audit logging **Troubleshooting** • Best practices for troubleshooting • Understanding the system limitations • Collecting details of the issues • Managing deadlocks • Installation and configuration issues • Upgrade issues • Network issues • File system issues • Disk issues • Security issues • Protocol issues • Disaster recovery issues • Performance issues • GUI issues • AFM issues • AFM DR issues • Transparent cloud tiering issues • File audit logging issues • Maintenance procedures • Recovery procedures • Support for troubleshooting • References	System administrators of GPFS systems who are experienced with the subsystems used to manage disks and who are familiar with the concepts presented in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Command and Programming Reference	This guide provides the following information: **Command reference** • gpfs.snap command • mmaddcallback command • mmadddisk command • mmaddnode command • mmadquery command • mmafmconfig command • mmafmctl command • mmafmlocal command • mmapplypolicy command • mmaudit command • mmauth command • mmbackup command • mmbackupconfig command • mmblock command • mmbuildgpl command • mmcallhome command • mmces command • mmcesdr command • mmchattr command • mmchcluster command • mmchconfig command • mmchdisk command • mmcheckquota command • mmchfileset command • mmchfs command • mmchlicense command • mmchmgr command • mmchnode command • mmchnodeclass command • mmchnsd command • mmchpolicy command • mmchpool command • mmchqos command • mmclidecode command • mmclone command • mmcloudgateway command • mmcrcluster command • mmcrfileset command • mmcrfs command • mmcrnodeclass command • mmcrnsd command • mmcrsnapshot command	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Command and Programming Reference	• mmdefedquota command • mmdefquotaoff command • mmdefquotaon command • mmdefragfs command • mmdelacl command • mmdelcallback command • mmdeldisk command • mmdelfileset command • mmdelfs command • mmdelnode command • mmdelnodeclass command • mmdelnsd command • mmdelsnapshot command • mmdf command • mmdiag command • mmdsh command • mmeditacl command • mmedquota command • mmexportfs command • mmfsck command • mmfsctl command • mmgetacl command • mmgetstate command • mmhadoopctl command • mmhealth command • mmimgbackup command • mmimgrestore command • mmimportfs command • mmkeyserv command • mmlinkfileset command • mmlsattr command • mmlscallback command • mmlscluster command • mmlsconfig command • mmlsdisk command	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Command and Programming Reference	• mmlsfileset command • mmlsfs command • mmlslicense command • mmlsmgr command • mmlsmount command • mmlsnodeclass command • mmlsnsd command • mmlspolicy command • mmlspool command • mmlsqos command • mmlsquota command • mmlssnapshot command • mmmigratefs command • mmmount command • mmmsgqueue command • mmnetverify command • mmnfs command • mmnsddiscover command • mmobj command • mmperfmon command • mmpmon command • mmprotocoltrace command • mmpsnap command • mmputacl command • mmquotaoff command • mmquotaon command • mmremotecluster command • mmremotefs command • mmrepquota command • mmrestoreconfig command • mmrestorefs command • mmrestripefile command • mmrestripefs command • mmrpldisk command • mmsdrrestore command • mmsetquota command • mmshutdown command • mmsmb command • mmsnapdir command • mmstartup command • mmtracectl command • mmumount command • mmunlinkfileset command • mmuserauth command • mmwinservctl command • spectrumscale command	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Command and Programming Reference	**Programming reference** • IBM Spectrum Scale Data Management API for GPFS information • GPFS programming interfaces • GPFS user exits • IBM Spectrum Scale management API commands	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users

IBM Spectrum Scale: Big Data and Analytics Guide	This guide provides the following information:
	
	IBM Spectrum Scale support for Hadoop
	• HDFS transparency
	• Supported IBM Spectrum Scale storage modes
	• Hadoop cluster planning
	• Installation and configuration of HDFS transparency
	• Application interaction with HDFS transparency
	• Upgrading the HDFS Transparency cluster
	• Rolling upgrade for HDFS Transparency
	• Security
	• Advanced features
	• Hadoop distribution support
	• Limitations and differences from native HDFS
	• Problem determination
	
	BigInsights® 4.2.5 and Hortonworks Data Platform 2.6
	• **Planning**
	  – Hardware requirements
	  – Preparing the environment
	  – Preparing a stanza file
	• **Installation**
	  – Set up
	  – Installation of software stack
	  – BigInsights value-add services on IBM Spectrum Scale
	• **Upgrading software stack**
	  – Migrating from BI IOP to HDP
	  – Upgrading IBM Spectrum Scale service MPack
	  – Upgrading HDFS Transparency
	  – Upgrading IBM Spectrum Scale file system
	  – Upgrading to BI IOP 4.2.5
		• System administrators of IBM Spectrum Scale systems
		• Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users

IBM Spectrum Scale: Big Data and Analytics Guide	• **Configuration**
	  – Setting up High Availability [HA]
	  – IBM Spectrum Scale configuration parameter checklist
	  – Dual-network deployment
	  – Manually starting services in Ambari
	  – Setting up local repository
	  – Configuring LogSearch
	  – Setting IBM Spectrum Scale configuration for BigSQL
	  – Hadoop Kafka/Zookeeper and IBM Spectrum Scale Kafka/Zookeeper
	• **Administration**
	  – IBM Spectrum Scale-FPO deployment
	  – Ranger
	  – Kerberos
	  – Short-circuit read (SSR)
	  – Disabling short circuit write
	  – IBM Spectrum Scale service management
	  – Ambari node management
	  – Restricting root access
	  – IBM Spectrum Scale management GUI
	  – IBM Spectrum Scale versus Native HDFS
	• **Troubleshooting**
	  – Snap data collection
	• **Limitations**
	  – Limitations and information
	• **FAQ**
	  – General
	  – Service fails to start
	  – Service check failures
		• System administrators of IBM Spectrum Scale systems
		• Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Prerequisite and related information

For updates to this information, see IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html).

For the latest support information, see the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Conventions used in this information

Table 2 describes the typographic conventions used in this information. UNIX file name conventions are used throughout this information.

Note: Users of IBM Spectrum Scale for Windows must be aware that on Windows, UNIX-style file names need to be converted appropriately. For example, the GPFS cluster configuration data is stored in the `/var/mmfs/gen/mmsdrfs` file. On Windows, the UNIX namespace starts under the `%SystemDrive%\cygwin64` directory, so the GPFS cluster configuration data is stored in the `C:\cygwin64\var\mmfs\gen\mmsdrfs` file.

Table 2. 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.
|	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.

Note: CLI options that accept a list of option values delimit with a comma and no space between values. As an example, to display the state on three nodes use **mmgetstate -N** *NodeA,NodeB,NodeC*. Exceptions to this syntax are listed specifically within the command.

How to send your comments

Your feedback is important in helping us to produce accurate, high-quality information. If you have any comments about this information or any other IBM Spectrum Scale documentation, send your comments to the following e-mail address:

mhvrcfs@us.ibm.com

Include the publication title and order number, and, if applicable, the specific location of the information about which you have comments (for example, a page number or a table number).

To contact the IBM Spectrum Scale development organization, send your comments to the following e-mail address:

gpfs@us.ibm.com

Summary of changes

This topic summarizes changes to the IBM Spectrum Scale licensed program and the IBM Spectrum Scale library. Within each information unit in the library, a vertical line (|) to the left of text and illustrations indicates technical changes or additions that are made to the previous edition of the information.

Summary of changes
for IBM Spectrum Scale version 5.0.0
as updated, November 2018

This release of the IBM Spectrum Scale licensed program and the IBM Spectrum Scale library includes the following improvements. All improvements are available after an upgrade, unless otherwise specified.

AFM and AFM DR-related changes

- Compression policy is supported. For more information, see *File compression* in *IBM Spectrum Scale: Administration Guide*.

- ILM for snapshots is supported. For more information, see *ILM for snapshots* in *IBM Spectrum Scale: Administration Guide*.

- Throttling the number of filesets to perform recoveries on the gateway node. For more information, see *Gateway node failure and recovery* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- Improved hashing algorithm to load balance filesets across gateway nodes. For more information, see *Primary gateway* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- Configuration parameters have been added - **afmMaxParallelRecoveries**, **afmAsyncOpWaitTimeout**, **afmSyncOpWaitTimeout**, and **afmRevalOpWaitTimeout**. For more information, see *Configuration parameters for AFM* and *Configuration parameters for AFM-based DR* in *IBM Spectrum Scale: Administration Guide*.

- Configuration parameters have been modified - **afmRPO** and **afmHashVersion**. For more information, see *Configuration parameters for AFM* and *Configuration parameters for AFM-based DR* in *IBM Spectrum Scale: Administration Guide*.

Authentication: Primary group selection configurable for AD + RFC2307 based authentication
Ability to choose primary group as set in "UNIX attributes" of a user on Active Directory introduced with AD + RFC2307 based authentication scheme. Earlier, Windows primary group was by default selected as the primary group.

Big data and analytics changes
For information on changes in IBM Spectrum Scale Big Data and Analytics support, see Big data and analytics - summary of changes.

IBM Spectrum Scale management API changes
Added the following new commands:

- GET /perfmon/data

- GET /filesystems/{filesystemName}/afm/state

- DELETE /nodes/{name}

- POST /nodes

- GET /nodeclasses

- POST /nodeclasses

- DELETE /nodeclasses/{nodeclassName}

- GET /nodeclasses/{nodeclassName}

- PUT /nodeclasses/{nodeclassName}
- DELETE /jobs/jobId
- POST /filesystems/{filesystemName}/filesets/{filesetName}/psnaps
- DELETE /filesystems/{filesystemName}/filesets/{filesetName}/psnaps/{snapshotName}
- GET /thresholds
- GET /thresholds/{name}
- POST /thresholds
- DELETE /thresholds/{name}

For more information on the API commands, see *IBM Spectrum Scale management API commands* in *IBM Spectrum Scale: Command and Programming Reference*. You can also access the documentation corresponding to each API command from the GUI itself. The API documentation is available in the GUI at: *https://<IP address or host name of API server>:<port>/ibm/api/explorer/*. For example: *https://scalegui.ibm.com:443/ibm/api/explorer*.

The JSON object structure is modified for the following API commands:

/cluster
> *rshSudoWrapper, rcpSudoWrapper*: Changed type from String to Boolean

/filesets
> Moved *filesetName* and *filesystemName* out of the *config* object.

/nfs/exports
> Removed the *config* object and moved the attributes one level up.

/nsds and /disks
> Removed the *config* object and moved the attributes one level up.
>
> *quorumDisk:* Changed type from String to Boolean.
>
> *size, availableBlocks, availableFragments*: Changed type from String to Long. Values are now in bytes.

/smb/shares
> Removed the *config* object and moved the attributes one level up.

/snapshots
> Removed the *config* object and moved the attributes one level up.

IBM Spectrum Scale GUI changes
> The following changes are made to the GUI:
>
> - Added new **Networks** page to monitor the performance, configuration, and adapters of network configurations in the cluster. You can monitor the network performance with respect to the IP and RDMA interfaces used in the configuration. For more information, see *Monitoring networks* in *IBM Spectrum Scale: Problem Determination Guide*.
>
> - Added new **Monitoring** > **Thresholds** page to create and monitor the threshold rules that are defined in the system. For more information, see *Monitoring thresholds* in *IBM Spectrum Scale: Problem Determination Guide*.
>
> - Added **Access** > **Remote Connections** page to enable the GUI node of the local cluster to monitor the remote cluster by establishing a connection with the GUI node of the remote cluster. For more information, see *Monitoring remote cluster through GUI* in *IBM Spectrum Scale: Problem Determination Guide*.
>
> - Added **Settings** > **Call Home** page to configure call home. Configuring the call home feature helps the IBM Support to monitor the system. Configuring call home also helps to reduce the response time of the IBM Support to resolve any issues.
>
>   The diagnostic data that is downloaded through the **Settings** > **Diagnostic Data** can be uploaded to a problem management record (PMR) by using the call home feature in the

backend. To upload the diagnostic data, right-click the relevant data set from the Previously Collected Diagnostic Data, and select **Upload to PMR**.For more information, see *Configuring call home using GUI* in *IBM Spectrum Scale: Problem Determination Guide*.

- Added the file system creation capabilities in GUI. Use the **Files** > **File Systems** > **Create File System** option to launch the Create File System wizard. In the Create File System wizard, you can specify the following details of the file system:
  - File system name
  - Storage pools
  - NSDs for the file systems
  - Failure groups
  - NSD order for data writes
  - Maximum number of Spectrum Scale clients
  - Maximum number of inodes of the root fileset
  - Whether to enable quota and scope for the quota definition
  - Whether to enable DMAPI
  - Mount point and automatic mount mode

  For more information, see *Creating and managing file systems using GUI* in *IBM Spectrum Scale: Administration Guide*.

- Added the aggregation levels *Access Point* and *Filesets* and removed *Account* for the resource type Transparent Cloud Tiering in the **Monitoring** > **Statistics** page.

- The **Files** > **Transparent Cloud Tiering** page now displays the file systems and filesets that are mapped with the TCT services. It also shows the connection of such a container pair configuration to a cloud account and the corresponding CSAP that is configured for the cloud account.

- Changed and updated information in the **Added DMPs for TIP events** page. The DMPs help the users resolve issues that are associated with the tip events. For more information, see the *Directed maintenance procedures for tip events* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

- **Changes to capacity monitoring in the GUI**

  The following changes are made to the capacity monitoring options in the GUI:
  - Moved the **Capacity** page from **Monitoring** to **Files** menu in the navigation and renamed the GUI page to **User Capacity**.
  - Only the file data user capacity can be viewed from the **Files** > **User Capacity** page. Removed the pools, filesets, file system capacity monitoring options from the **Files** > **User Capacity** page. You can monitor the capacity of these components from the respective GUI pages.
  - Replaced the GPFSPoolCap sensor with the GPFSPool sensor and separate data and metadata level capacity monitoring are introduced in the performance charts available in the **Files** > **File Systems** and **Storage** > **Pools** pages.
  - New GPFSPool-based data and metadata performance monitoring metrics are available for selection in the **Files** > **Statistics** > **Edit** > **Capacity** section. You need to select the aggregation level as Pool to view these metrics.

  For more information, see *Monitoring capacity through GUI* in *IBM Spectrum Scale: Problem Determination Guide*.

- **AFM monitoring changes in the GUI**

  The following AFM monitoring features are introduced:
  - Provides the number of AFM filesets and the corresponding export server maps. Each export map establishes a mapping between the gateway node and the NFS host name to allow parallel data transfers from cache to home.

– By using the **Request Access** option available in the **Files** > **Active File Management** or **Access** > **Remote Connection** page in the GUI, you can now establish connection with remote clusters. After establishing the connection, you can monitor the following AFM and AFM DR configuration details across clusters:

- On home and secondary, you can see the AFM relationships configuration, health status, and performance values of the Cache and Disaster Recovery grids.
- On the Overview tab of the detailed view, the available home and secondary inodes are available.
- On the Overview tab of the detailed view, the details such as NFS throughput, IOPs, and latency details are available, if the protocol is NFS.

For more information, see *Monitoring AFM and AFM DR using GUI* in *IBM Spectrum Scale: Problem Determination Guide*.

- New option to create AFM peer snapshots through GUI. Use the **Create Peer Snapshot** option in the **Files** > **Snapshots** page to create peer snapshots. You can view and delete these peer snapshots from the Snapshots page and also from the detailed view of the **Files** > **Active File Management** page.

IBM Spectrum Scale functionality to support GDPR requirements

To understand the requirements of EU General Data Protection Regulation (GDPR) compliance that are applicable to unstructured data storage and how IBM Spectrum Scale helps to address them, see the IBM Spectrum Scale functionality to support GDPR requirements technote.

Encryption: GSKit V8 improves cryptographic performance on IBM POWER8®

The IBM Global Security Kit (GSKit) Version 8 and later improves cryptographic performance on IBM POWER8 hardware. For more information, see the IBM article *GSKIT V8 Support for PKCS#11 device integration and on-CPU acceleration* (https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W746177d414b9_4c5f_9095_5b8657ff8e9d/page/PKCS%2311%20device%20integration). The version of GSKit that is shipped with IBM Spectrum Scale v5.0.0 offers better performance on POWER8, compared with the versions shipped with earlier releases.

File audit logging

File audit logging is a new capability in IBM Spectrum Scale v5.0.0. File audit logging takes locally generated file system events and puts them on a multinode message queue from which they are consumed and written to a retention enabled fileset. These events, called lightweight events, occur at the file system level. They are protocol agnostic, which means that they capture all access to a monitored file system from protocol exports to even root access that occurs directly on nodes. File audit logging is integrated into the system health infrastructure. Alerts are generated for elements of the message queue and the processes that consume the events and create the audit logs. For more information, see *Introduction to file audit logging* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

File compression: The lz4 library provides fast access to compressed data

File compression supports the lz4 compression library. Lz4 is intended primarily for active data and favors read-access speed over maximized space saving. For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration Guide*.

File data: Smaller subblock sizes improve I/O performance and reduce fragmentation

Subblock sizes are smaller relative to block sizes; for example, a 4 MiB block uses 8 KiB subblocks. The same choice of data block sizes is available, but the default data block size is larger, 4 MiB instead of 256 KiB. A larger block size improves the file system performance and a smaller subblock size reduces the amount of unused space. For many business applications, the default value of 4 MiB provides the best balance of improved performance and reduced fragmentation. For more information, see *mmcrfs command* in the *IBM Spectrum Scale: Command and Programming Reference*. This feature is supported only for file systems that are created at file system format number 18.00 (IBM Spectrum Scale 5.0.0.0) or later.

File encryption: AES-XTS encryption is faster on x86 in non-FIPS mode

On x86 architecture in non-FIPS mode, file encryption with the AES algorithm in XTS mode is faster than it was. For more information, see the topics *Encryption policy rules* and *Encryption and FIPS-140-2 certification* in the *IBM Spectrum Scale: Administration Guide*.

File systems: File system rebalancing is faster

Rebalancing is implemented by a lenient round-robin method that typically runs faster than the previously used method of strict round robin. The strict round robin method is available as an option. For more information, see the topics *mmrestripefs command*, *mmrestripefile command*, *mmadddisk command*, and *mmdeldisk command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Installation toolkit changes

- The installation toolkit has added support for the installation and the deployment of protocol and client nodes in a cluster that contains Elastic Storage Server (ESS). For more information, see *ESS awareness with the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- The installation toolkit has added support for enabling and configuring call home. For more information, see *Enabling and configuring call home using the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- The installation toolkit has added support for enabling and configuring file audit logging. For more information, see *Enabling and configuring file audit logging using the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- The installation toolkit has added support for the installation and the deployment of IBM Spectrum Scale on Ubuntu 16.04 LTS nodes.

- The installation toolkit has added support for the installation and the deployment of IBM Spectrum Scale on Red Hat Enterprise Linux 7.4 nodes.

- The installation toolkit has added verification of passwordless SSH during prechecks before installation, deployment, or upgrade.

- The installation toolkit has added support for cumulative object upgrade.

mmafmctl command

The **--outband** parameter is deprecated.

mmcallhome command: Enhancements

Addition of -Y option

The **-Y** displays the command output in a parseable format with a colon (:) as a field delimiter.

Addition of --pmr option

The **--pmr** option allows you to upload data to existing PMRs using the **mmcallhome run SendFile** command. For more information, see *mmcallhome command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmchconfig command: Enhancements

Encrypted files can be copied into an LROC device

With the **lrocEnableStoringClearText** attribute, you can control whether file data from encrypted files, which is held in memory as cleartext, is copied into a local read-only cache (LROC) device.

InfiniBand addresses can be specified for RDMA transfers

In the **verbsPorts** attribute, you can specify InfiniBand addresses for RDMA transfers between an NSD client and server.

For more information, see *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmchnsd command: Change NSDs without unmounting the file system

When you add or remove NSDs or do other operations with `mmchnsd`, you do not need to unmount the file system. For more information, see the topic *mmchnsd command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmcrfs command: Enhancements

The default data block size is 4 MiB with an 8 KiB subblock size

If no block size is specified, a file system is created with a 4 MiB block size and an 8 KiB subblock size. The minimum release level (`minReleaseLevel`) of the cluster must be 5.0.0 or greater when the file system is created. For more information, see the entry **File data: Smaller subblock sizes improve I/O performance and reduce fragmentation** earlier in this topic.

The default log file size depends on block size and metadata size

If the block size is 512 KiB or larger and the metadata block size 256 KiB or larger, then the default log file size is 32 MiB. Otherwise, the default log file size is 4 MiB or the metadata block size, whichever is larger. For more information, see the topic *mmcrfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

The default method for updating `atime` is `relatime`

If the minimum release level (`minReleaseLevel`) of the cluster is 5.0.0 or greater when the file system is created, the default method for updating `atime` is `relatime`. For more information, see the topics *mmcrfs command* and *mmchfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmdsh command: Several options are no longer supported

The `--ignoreSignal`, `-I`, and `-d` options are no longer supported. Do not use these options unless instructed to by IBM support personnel. For more information, see the topic *mmdsh command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmfsck command: Display an interim status report at any time

While a long-running instance of `mmfsck` is in progress, you can start another instance of `mmfsck` with the `--status-report` parameter to display current status information from all the nodes that are participating in the `mmfsck` run. For more information, see the topic *mmfsck command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmgetstate command: Display the `unresponsive` state

The command returns the **unresponsive** state when the GPFS daemon is running but is not responding. For more information, see the topic *mmgetstate command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmhealth command: Addition to measurement options

Measurement options for filesystem, SMB node, and NFS node has been added to the `mmhealth` command. For more information, see the topic *mmhealth command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmkeyserv command: The simplified setup method supports certificate chains from a certificate authority.

In the simplified setup, with the `--kmip-cert` parameter, you can set up encryption with IBM® Security Key Lifecycle Manager (SKLM) as the key management server and with a certificate signed by a certificate authority (CA) on the KMIP port of the SKLM server. For more information, see the topic *mmkeyserv command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmnetverify command: Enhancements

Verify the network operation of nodes in a subnet

With the `--subnets` parameters, you can specify the subnet addresses of the nodes that you want to verify.

Verify that nodes can handle a new MTU size

With the `ping-packet-size` parameter, you can specify the size of the ICMP echo request packets that are sent between local node and the target node during the `ping` test.

For more information, see *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

mmpmon command: Enhancements

A new option `loc_io_s` is added to `mmpmon` command. It stands for locality I/O statistics.

mmtracectl command: Display the tracing status

The `--status` parameter displays the tracing status of the specified nodes. For more information, see *mmtracectl command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Threshold monitoring enhancements

If multiple thresholds rules have overlapping entities for the same metrics, only one of the concurrent rules is made actively eligible. For more information, see the *New features for threshold monitoring* section in *Threshold monitoring for system health* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

NFS: Dynamic export changes

You can dynamically change the export configuration without restarting the NFS service. For more information, see the *Changing NFS export configuration* topic in the *IBM Spectrum Scale: Administration Guide*.

Object changes

The following changes have been done.

- Object protocol support is extended to Ubuntu 16.04 LTS.
- The sudo wrapper for object is added on Ubuntu 16.04 LTS.
- Cumulative object upgrades from older versions are supported.
- Object snap enhancement are done to contain Keystone logs.

Protocol support: Enhanced

NFS, SMB, and object support is extended to Ubuntu 16.04 LTS.

Setting up a system for storing crash files for Ubuntu

Support is added to set up a system for storing crash files for Ubuntu. For more information, see the *Setting up an Ubuntu system to capture crash files* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

SMB: DFS redirects for SMB shares

New option to configure DFS redirects for SMB shares. For more information, see *mmsmb command* in the *IBM Spectrum Scale: Command and Programming Reference*.

SMB: SMB server upgrade changes

Two events on CTDB version match/mismatch are added to the RAS events. For more information, see the *RAS events* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

Sudo wrappers: Root-level processes can call administration commands directly

Root-level background processes, such as `cron` and callback programs, can successfully call administration commands directly rather than through `sudo` when sudo wrappers are enabled. For more information, see *Root-level processes that call administration commands directly* in the *IBM Spectrum Scale: Administration Guide*.

Cloud services changes

The following enhancements are made:

- A new robust set of CLI functions, which allow for much greater configuration of the system.
- Support for multiple cloud storage accounts
- Support for multiple file systems or filesets per node group

- Enhanced support for large file systems provided by container spillover
- Support for associating file sets with containers for enhanced granularity
- Multiple URL and region support at the node level
- Support for creating a Cloud services separately for tiering and sharing operations. For more information, see the *Creating Cloud services* topic in the *IBM Spectrum Scale: Administration Guide*.
- Unique encryption key per cloud container
- Support for remotely mounted clients. For more information, see the *Setting up Transparent cloud tiering service on a remotely mounted client* topic in the *IBM Spectrum Scale: Administration Guide*.
- Support for Amazon S3 regions requiring Sigv4 security support, including the US government cloud region.
- Ability to enable or disable transparent recall for files for a given file system instantly, without having to rewrite a policy.
- Support for backing up and restoring the Cloud services configuration in case of any disaster. For more information, see the *Backing up the Cloud services configuration* topic in the *IBM Spectrum Scale: Administration Guide*.
- Support for backing up the Cloud services database to the cloud. For more information, see the *Backing up the Cloud services database to the cloud* topic in the *IBM Spectrum Scale: Administration Guide*.
- Support for restoring Transparent cloud tiering service on an identical backup cluster. For more information, see the *Restoring Transparent Cloud services on a backup cluster* topic in the *IBM Spectrum Scale: Administration Guide*.
- Support for checking the integrity of the Cloud services database after any system crash or outage. For more information, see the *Checking the Cloud services database integrity* topic in the *IBM Spectrum Scale: Administration Guide*.
- Support for auditing events relating to each operation performed in Cloud services. For more information, see the *Cloud services audit events* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

Documented commands, structures, and subroutines

The following lists the modifications to the documented commands, structures, and subroutines:

New commands

These commands are new in this release.

- **mmaudit**
- **mmmsgqueue**

New structures

There are no new structures.

New subroutines

There are no new subroutines.

Changed commands

The following commands were changed:

- **mmafmctl**
- **mmadddisk**
- **mmcallhome**
- **mmchattr**
- **mmchcluster**
- **mmchconfig**
- **mmchfs**
- **mmchnsd**

- **mmcloudgateway**
- **mmcrcluster**
- **mmcrfs**
- **mmdeldisk**
- **mmdsh**
- **mmfsck**
- **mmgetstate**
- **mmkeyserv**
- **mmnetverify**
- **mmnfs**
- **mmpmon**
- **mmrestripefile**
- **mmrestripefs**
- **mmsmb**
- **mmtracectl**
- **mmuserauth**
- **spectrumscale**

Changed structures
> There are no changed structures.

Changed subroutines
> There are no changed subroutines.

Deleted commands
> **mmrest**

Deleted structures
> There are no deleted structures.

Deleted subroutines
> There are no deleted subroutines.

Messages
> The following are the new, changed, and deleted messages:

New messages
> 6027-1264, 6027-1757, 6027-2394, 6027-2395, 6027-2396, 6027-2397, 6027-2398, 6027-2399,
> 6027-2400, 6027-2401, 6027-3259, 6027-3408, 6027-3597, 6027-3598, 6027-3599, 6027-3600,
> 6027-3601, 6027-3602, 6027-3603, 6027-3604, 6027-3730, 6027-3921, 6027-3922, 6027-3923,
> 6027-3924, 6027-3925, 6027-3926, 6027-3927, 6027-3928, 6027-3929, 6027-3930, 6027-3931,
> and 6027-4019.

Changed messages
> 6027-928

Deleted messages
> None.

Changes in documentation

Separate sections for installing and upgrading documentation
> The *Installing and upgrading* section is split into two separate sections:
> - *Installing*
> - *Upgrading*

Terminology change in upgrade documentation

In the upgrade documentation, the instances of "migration" are replaced with "upgrade".

Product name changes

The following product name changes are done:

- z System is renamed to IBM Z
- Linux on IBM Z is renamed to Linux on Z

List of documentation changes in product guides and respective Knowledge Center sections

The following is a list of documentation changes including changes in topic titles, changes in placement of topics, and deleted topics:

Table 3. List of changes in documentation

Guide	Knowledge center section	List of changes
Concepts, Planning, and Installation Guide	Product overview	The following sub-sections are moved from the *Introduction to Cloud services* topic to *Library and related publications → Applying IBM Spectrum Scale*: • *Transparent cloud tiering use cases* • *Supported Cloud data sharing use cases* • *Supported WORM storage use cases*

	Installing	• *Creating a user-defined node class for Transparent cloud tiering or Cloud data sharing* topic is moved from the *Configuring* section to the *Installing* section
		• The *IBM Spectrum Scale packaging overview* topic is moved one level up.
		• The following topics and their sub-topics are removed. The content in these topics is now covered in *Manually installing IBM Spectrum Scale and deploying protocols on Linux nodes* and its sub-topics.
		  – *Manually installing IBM Spectrum Scale on Red Hat Enterprise Linux 7.x systems*
		  – *Manually installing IBM Spectrum Scale on SLES 12 systems*

	Upgrading	The upgrade topics have been reordered to bring out the upgrade flow more clearly and separate manual and installation toolkit topics clearly.
Administration Guide	Configuring	The following topics are removed from the *Configuring and tuning your system for Cloud services* section: • *Pre-validating the cloud account settings* • *Associating a file system with Cloud services nodes*
	Administering	Moved the *Manual recovery of Transparent cloud tiering database* topic under *Administering files for Transparent cloud tiering* section

Changes in the Library and related publications section

A *Video resources* topic was added in this section. This topic contains links to IBM Spectrum Scale videos.

Chapter 1. Configuring the GPFS cluster

There are several tasks involved in managing your GPFS cluster. This topic points you to the information you need to get started.

GPFS cluster management tasks include the following.
- "Creating your GPFS cluster"
- "Displaying cluster configuration information"
- "Specifying nodes as input to GPFS commands" on page 103
- "Adding nodes to a GPFS cluster" on page 2
- "Deleting nodes from a GPFS cluster" on page 3
- "Changing the GPFS cluster configuration data" on page 4
- "Node quorum considerations" on page 20
- "Node quorum with tiebreaker considerations" on page 20
- "Displaying and changing the file system manager node" on page 21
- "Determining how long `mmrestripefs` takes to complete" on page 21
- "Starting and stopping GPFS" on page 22

Note: In IBM Spectrum Scale V4.1.1 and later, many of these tasks can also be handled by the installation toolkit configuration options. For more information on the installation toolkit, see *Using the spectrumscale installation toolkit to perform installation tasks: Explanations and examples* topic in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For information on RAID administration, see *IBM Spectrum Scale RAID: Administration*.

Creating your GPFS cluster

You must first create a GPFS cluster by issuing the **mmcrcluster** command.

For more information, see **mmcrcluster command** in *IBM Spectrum Scale: Command and Programming Reference*.

For details on how GPFS clusters are created and used, see *GPFS cluster creation considerations* topic in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Displaying cluster configuration information

Use the **mmlscluster** command to display cluster configuration information.
"Basic configuration information"
"Information about protocol nodes" on page 2

For more usage information, see **mmlscluster command** in *IBM Spectrum Scale: Command and Programming Reference*.

Basic configuration information

To display basic cluster configuration information, enter the **mmcluster** command with no parameters:
```
mmlscluster
```

The command displays information like the following example:

```
GPFS cluster information
========================
  GPFS cluster name:         cluster1.kgn.ibm.com
  GPFS cluster id:           680681562214606028
  GPFS UID domain:           cluster1.kgn.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

  Node  Daemon node name       IP address       Admin node name          Designation
-----------------------------------------------------------------------------------------
    1   k164n04.kgn.ibm.com    198.117.68.68    k164n04.kgn.ibm.com      quorum
    2   k164n05.kgn.ibm.com    198.117.68.69    k164n05.kgn.ibm.com      quorum
    3   k164n06.kgn.ibm.com    198.117.68.70    k164sn06.kgn.ibm.com     quorum-manager
```

If the cluster uses a server-based repository, the command also displays the following information:

- The primary GPFS cluster configuration server
- The secondary GPFS cluster configuration server

Information about protocol nodes

To display information about the protocol nodes, enter the following command:

```
mmlscluster --ces
```

The command displays information like the following example:

```
GPFS cluster information
========================
GPFS cluster name: cluster1.kgn.ibm.com
GPFS cluster id: 4708497829760395040

Cluster Export Services global parameters
-----------------------------------------
Shared root directory: /gpfs/ces/ces
Enabled Services: OBJ SMB NFS
Log level: 0
Address distribution policy: even-coverage

Node Daemon node name IP address CES IP address list
----------------------------------------------------------------
4 k16n07.kgn.ibm.com 192.168.4.4 10.18.64.23
5 k16n08.kgn.ibm.com 192.168.4.5 10.18.64.24
6 k16n09.kgn.ibm.com 192.168.4.6 10.18.64.26
7 k16n10.kgn.ibm.com 192.168.4.11 Node suspended, Node starting up
8 k16n11.kgn.ibm.com 192.168.4.12 Node suspended, Node starting up
```

Adding nodes to a GPFS cluster

You can add nodes to an existing GPFS cluster by issuing the **mmaddnode** command. The new nodes are available immediately after the successful completion of this command.

You must follow these rules when adding nodes to a GPFS cluster:

- You may issue the command only from a node that already belongs to the GPFS cluster.
- A node may belong to only one GPFS cluster at a time.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command to add those nodes.
- After the nodes are added to the cluster, you must use the **mmchlicense** command to designate appropriate GPFS licenses to the new nodes.

To add node k164n01.kgn.ibm.com to the GPFS cluster, issue the following command:

```
mmaddnode -N k164n01.kgn.ibm.com
```

The system displays information similar to the following:

```
Mon Aug  9 21:53:30 EDT 2004: 6027-1664 mmaddnode: Processing node k164n01.kgn.ibm.com
mmaddnode: Command successfully completed
mmaddnode: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To confirm the addition of the nodes, issue the following command:

```
mmlscluster
```

The command displays information similar to the following:

```
GPFS cluster information
========================
  GPFS cluster name:         cluster1.kgn.ibm.com
  GPFS cluster id:           15529849231188177215
  GPFS UID domain:           cluster1.kgn.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name     IP address        Admin node name        Designation
--------------------------------------------------------------------------------
    1  k164n01.kgn.ibm.com  198.117.68.66     k164n01.kgn.ibm.com
    2  k164n02.kgn.ibm.com  198.117.68.67     k164n02.kgn.ibm.com
    3  k164n03.kgn.ibm.com  198.117.68.68     k164n03.kgn.ibm.com    quorum
    4  k164n04.kgn.ibm.com  198.117.68.69     k164n04.kgn.ibm.com    quorum
    3  k164n05.kgn.ibm.com  198.117.68.70     k164n05.kgn.ibm.com    quorum-manager
```

You can also use the installation toolkit to add nodes. For more information, see *Adding nodes, NSDs, or file systems to an installation process* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For complete usage information, see `mmaddnode command`, `mmlscluster command` and `mmchlicense command` in *IBM Spectrum Scale: Command and Programming Reference*.

Deleting nodes from a GPFS cluster

You can delete nodes from a GPFS cluster by issuing the **mmdelnode** command.

You must follow these rules when deleting nodes:
- The GPFS daemon must be shut down on the nodes being deleted. Issue the **mmshutdown** command.
- A node that is being deleted cannot be designated as an NSD server for any disk in the GPFS cluster, unless you intend to delete the entire cluster. Verify this by issuing the **mmlsnsd** command. If a node that is to be deleted is an NSD server for one or more disks, move the disks to nodes that will remain in the cluster. Issue the **mmchnsd** command to assign new NSD servers for those disks.
- If your cluster has a traditional server-based (non-CCR) configuration repository, you cannot delete a primary or secondary cluster configuration server, unless you intend to delete the entire cluster. To determine whether the node is a cluster configuration server, issue the **mmlscluster** command. If the node to be deleted is one of the servers and you intend to keep the cluster, issue the **mmchcluster** command to assign another node as a configuration server before you delete the node.

To delete the nodes listed in a file called **nodes_to_delete**, issue the following command:

```
mmdelnode -N /tmp/nodes_to_delete
```

where **nodes_to_delete** contains the nodes k164n01 and k164n02. The system displays information similar to the following:

```
Verifying GPFS is stopped on all affected nodes ...
mmdelnode: Command successfully completed
mmdelnode: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To confirm the deletion of the nodes, issue the following command:

```
mmlscluster
```

The system displays information similar to following:

```
GPFS cluster information
========================
  GPFS cluster name:         cluster1.kgn.ibm.com
  GPFS cluster id:           15529849231188177215
  GPFS UID domain:           cluster1.kgn.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name       IP address      Admin node name        Designation
-----------------------------------------------------------------------------------
   1   k164n03.kgn.ibm.com    198.117.68.68   k164n03.kgn.ibm.com    quorum
   2   k164n04.kgn.ibm.com    198.117.68.69   k164n04.kgn.ibm.com    quorum
   3   k164n05.kgn.ibm.com    198.117.68.70   k164n05.kgn.ibm.com    quorum-manager
```

For information on deleting protocol nodes (CES nodes) from a cluster, see "Deleting a Cluster Export Services node from an IBM Spectrum Scale cluster" on page 33.

For complete usage information, see **mmdelnode command** and **mmlscluster command** in *IBM Spectrum Scale: Command and Programming Reference*.

Exercise caution when shutting down GPFS on quorum nodes or deleting quorum nodes from the GPFS cluster. If the number of remaining quorum nodes falls below the requirement for a quorum, you will be unable to perform file system operations. For more information on quorum, see the section on *Quorum*, in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Changing the GPFS cluster configuration data

You can use the **mmchcluster** or **mmchconfig** commands to change the configuration attributes.

After you have configured the GPFS cluster, you can change configuration attributes with the **mmchcluster** command or the **mmchconfig** command. For more information, see the following topics:
* *mmchcluster command* in *IBM Spectrum Scale: Command and Programming Reference*
* *mmchconfig command* in *IBM Spectrum Scale: Command and Programming Reference*

Use the **mmchcluster** command to do the following tasks:
* Change the name of the cluster.
* Change the remote shell and remote file copy programs to be used by the nodes in the cluster. These commands must adhere to the syntax forms of the **ssh** and **scp** commands, but may implement an alternate authentication mechanism.
* Enable or disable the cluster configuration repository (CCR). For more information, see the *Cluster configuration data files* topic in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

If you are using the traditional server-based (non-CCR) configuration repository, you can also do the following tasks:
* Change the primary or secondary GPFS cluster configuration server nodes. The primary or secondary server may be changed to another node in the GPFS cluster. That node must be available for the command to be successful.

Attention: If during the change to a new primary or secondary GPFS cluster configuration server, one or both of the old server nodes are down, it is imperative that you run the **mmchcluster -p LATEST** command as soon as the old servers are brought back online. Failure to do so may lead to disruption in GPFS operations.

- Synchronize the primary GPFS cluster configuration server node. If an invocation of the **mmchcluster** command fails, you will be prompted to reissue the command and specify **LATEST** on the **-p** option to synchronize all of the nodes in the GPFS cluster. Synchronization instructs all nodes in the GPFS cluster to use the most recently specified primary GPFS cluster configuration server.

For example, to change the primary server for the GPFS cluster data, enter:

```
mmchcluster -p k164n06
```

The system displays information similar to:

```
mmchcluster -p k164n06
mmchcluster: Command successfully completed
```

To confirm the change, enter:

```
mmlscluster
```

The system displays information similar to:

```
GPFS cluster information
========================
  GPFS cluster name:         cluster1.kgn.ibm.com
  GPFS cluster id:           680681562214606028
  GPFS UID domain:           cluster1.kgn.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           server-based

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    k164sn06.kgn.ibm.com
  Secondary server:  k164n05.kgn.ibm.com

  Node  Daemon node name      IP address      Admin node name        Designation
--------------------------------------------------------------------------------
    1   k164n04.kgn.ibm.com   198.117.68.68   k164n04.kgn.ibm.com    quorum
    2   k164n05.kgn.ibm.com   198.117.68.69   k164n05.kgn.ibm.com    quorum
    3   k164n06.kgn.ibm.com   198.117.68.70   k164sn06.kgn.ibm.com   quorum-manager
```

Attention: The **mmchcluster** command, when issued with either the **-p** or **-s** option, is designed to operate in an environment where the current primary and secondary GPFS cluster configuration servers are *not* available. As a result, the command can run without obtaining its regular serialization locks. To assure smooth transition to a new cluster configuration server, no other GPFS commands (**mm...** commands) should be running when the command is issued nor should any other command be issued until the **mmchcluster** command has successfully completed.

For complete usage information, see **mmchcluster command** and **mmlscluster command** in *IBM Spectrum Scale: Command and Programming Reference*

You might be able to tune your cluster for better performance by reconfiguring one or more attribute. Before you change any attribute, consider how the changes will affect the operation of GPFS. For a detailed discussion, see *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* and *mmcrcluster command* in *IBM Spectrum Scale: Command and Programming Reference* guide.

Table 4 on page 6 details the GPFS cluster configuration attributes which can be changed by issuing the **mmchconfig** command. Variations under which these changes take effect are noted:

1. Take effect immediately and are permanent (**-i**).

2. Take effect immediately but do not persist when GPFS is restarted (**-I**).
3. Require that the GPFS daemon be stopped on all nodes for the change to take effect.
4. May be applied to only a subset of the nodes in the cluster.

For more information on the release history of tuning parameters, see "Tuning parameters change history" on page 48.

Table 4. Configuration attributes on the **mmchconfig** *command*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeNames* allowed	Change takes effect
adminMode Controls password-less access	yes	no	no	no	immediately
atimeDeferredSeconds Update behavior of **atime** when **relatime** is enabled	yes	yes	no	yes	if not immediately, on restart of the daemon
autoload Starts GPFS automatically	no	no	no	yes	on reboot of each node
automountDir Name of the automount directory	no	no	yes	no	on restart of the daemon
cesSharedRoot A directory to be used by the CES subsystem.	no	no	yes (on all CES nodes)	no	immediately
cipherList The security mode of the cluster. This value indicates the level of security that the cluster uses for communications between nodes in the cluster and also for communications between clusters.	no	no	only when changing from `AUTHONLY` or a cipher to `EMPTY` mode	no	for new connections
cnfsGrace The number of seconds a CNFS node will deny new client requests after a node failover or failback	yes	no	yes	no	immediately
cnfsMountdPort The port number to be used for **rpc.mountd**	yes	no	no	no	immediately
cnfsNFSDprocs The number of **nfsd** kernel threads	yes	no	no	no	if not immediately, on restart of the daemon
cnfsReboot Determines whether the node will reboot when CNFS monitoring detects an unrecoverable problem.	yes	no	no	yes	immediately
cnfsSharedRoot Directory to be used by the clustered NFS subsystem	yes	no	yes	no	immediately

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
cnfsVersions List of protocol versions that CNFS should start and monitor	yes	no	yes	no	immediately
dataDiskCacheProtectionMethod Defines the cache protection method for disks that are used for the GPFS file system.	no	no	yes	no	on restart of the daemon
dataDiskWaitTimeForRecovery Controls the suspension of **dataOnly** disk recovery	yes	no	no	yes	immediately
dataStructureDump Path for the storage of dumps	yes	no	no	yes	if not immediately, on restart of the daemon
deadlockBreakupDelay When to attempt breaking up a detected deadlock	yes	yes	no	no	immediately with **-i** or **-I**
deadlockDataCollectionDailyLimit Maximum number of times to collect debug data in 24 hours	yes	yes	no	no	immediately with **-i** or **-I**
deadlockDataCollectionMinInterval Minimum interval between two consecutive collections of debug data	yes	yes	no	no	immediately with **-i** or **-I**
deadlockDetectionThreshold Threshold for detecting deadlocks	yes	yes	no	no	immediately with **-i** or **-I**
deadlockDetectionThresholdForShortWaiters Threshold for detecting deadlocks from short waiters	yes	yes	no	no	immediately with **-i** or **-I**
deadlockDetectionThresholdIfOverloaded Threshold for detecting deadlocks when a cluster is overloaded	yes	yes	no	no	immediately with **-i** or **-I**
deadlockOverloadThreshold Threshold for detecting cluster overload	yes	yes	no	no	immediately with **-i** or **-I**
debugDataControl Controls the amount of debug data collected	yes	no	no	yes	immediately
defaultMountDir Default parent directory for GPFS file systems	yes	yes	no	no	for new file systems

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
disableInodeUpdateOnFdatasync Controls inode update on **fdatasync** for **mtime** and **atime** updates.	yes	yes	no	yes	immediately with **-i** or **-I**
dmapiDataEventRetry DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
dmapiEventTimeout DMAPI attribute	no	no	no	yes	on restart of the daemon
dmapiMountEvent DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
dmapiMountTimeout DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
dmapiSessionFailureTimeout DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
enableIPv6 Controls whether the GPFS daemon is to communicate through the IPv6 network.	no	no	only when **enableIPv6** is set to **yes**	not applicable	if not immediately, on restart of the daemon
enforceFilesetQuotaOnRoot Controls fileset quota settings for the root user	yes	yes	no	no	if not immediately, on restart of the daemon
expelDataCollectionDailyLimit Maximum number of times to collect expel-related debug data in 24 hours	yes	yes	no	no	immediately with **-i** or **-I**
expelDataCollectionMinInterval Minimum interval between two consecutive collections of expel-related debug data	yes	yes	no	no	immediately with **-i** or **-I**
failureDetectionTime Indicates the amount of time it will take to detect that a node has failed	no	no	yes	no	on restart of the daemon
fastestPolicyCmpThreshold Indicates the disk comparison count threshold, above which GPFS forces selection of this disk as the preferred disk to read	yes	yes	no	yes	immediately with -i

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
fastestPolicyMaxValidPeriod Indicates the time period after which the disk's current evaluation is considered invalid	yes	yes	no	yes	immediately with -i
fastestPolicyMinDiffPercent A percentage value indicating how GPFS selects the fastest between two disks	yes	yes	no	yes	immediately with -i
fastestPolicyNumReadSamples Controls how many read samples taken to evaluate the disk's recent speed	yes	yes	no	yes	immediately with -i
fileHeatLossPercent Specifies the reduction rate of **FILE_HEAT** value for every **fileHeatPeriodMinutes** of file inactivity.	yes	yes	no	no	if not immediately, on restart of the daemon
fileHeatPeriodMinutes Specifies the inactivity time before a file starts to lose **FILE_HEAT** value.	yes	yes	no	no	if not immediately, on restart of the daemon
FIPS1402mode Controls whether GPFS operates in FIPS 140-2 mode.	no	no	no	not applicable	on restart of the daemon
forceLogWriteOnFdatasync Controls forcing log writes to disk.	yes	yes	no	yes	immediately with **-i** or **-I**
ignorePrefetchLUNCount The GPFS client node calculates the number of sequential access prefetch and write-behind threads to run concurrently for each file system by using the count of the number of LUNs in the file system and the value of **maxMBpS**.	yes	yes	no	yes	immediately with **-i**
lrocData Controls whether user data will be populated into the local read-only cache.	yes	yes	no	yes	immediately with **-i** or **-I**
lrocDataMaxFileSize Limits the data that may be saved in the local read-only cache to only the data from small files.	yes	yes	no	yes	immediately with **-i** or **-I**
lrocDataStubFileSize Limits the data that may be saved in the local read-only cache to only the data from the first portion of all files.	yes	yes	no	yes	immediately with **-i** or **-I**

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
lrocDirectories Controls whether directory blocks will be populated into the local read-only cache.	yes	yes	no	yes	immediately with **-i** or **-I**
lrocEnableStoringClearText Controls whether encrypted file data can be read into a local read-only cache (LROC) device.	yes	yes	no	no	immediately with **-i** or **-I**
lrocInodes Controls whether inodes from open files will be populated into the local read-only cache.	yes	yes	no	yes	immediately with **-i** or **-I**
maxblocksize Maximum file system block size allowed	no	no	no	yes	on restart of the daemon
maxBufferDescs Can be tuned to cache very large files	no	no	no	yes	on restart of the daemon
maxDownDisksForRecovery Maximum number of failed disks allowed for automatic recovery to continue	yes	no	no	yes	immediately
maxFailedNodesForRecovery Maximum number of unavailable nodes allowed before automatic disk recovery is cancelled	yes	no	no	yes	immediately
maxFcntlRangesPerFile Specifies the number of **fcntl** locks that are allowed per file	yes	yes	no	yes	if not immediately, on restart of the daemon
maxFilesToCache Number of inodes to cache for recently used files	no	no	no	yes	on restart of the daemon
maxMissedPingTimeout Handles high network latency in a short period of time	no	no	no	no	on restart of the daemon
maxMBpS I/O throughput estimate	yes	yes	no	yes	if not immediately, on restart of the daemon
maxStatCache Number of inodes to keep in stat cache	no	no	no	yes	on restart of the daemon
metadataDiskWaitTimeForRecovery Controls the suspension of metadata disk recovery	yes	no	no	yes	immediately

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
minDiskWaitTimeForRecovery Controls the suspension of disk recovery	yes	no	no	yes	immediately
minMissedPingTimeout Handles high network latency in a short period of time	no	no	no	no	on restart of the daemon
mmapRangeLock Specifies POSIX or non-POSIX **mmap** byte-range semantics **Note:** The list of *NodeName*s is allowed, but it is not recommended.	yes	yes	no	yes	immediately
nfsPrefetchStrategy Optimizes prefetching for NFS file-style access patterns	yes	yes	no	yes	immediately with **-i**
nistCompliance Controls whether GPFS operates in NIST 800-131A mode for security transport mechanisms.	no	no	no	not applicable	if not immediately, on restart of the daemon
noSpaceEventInterval Time interval between **noDiskSpace** events of a file system	yes	yes	no	yes	if not immediately, on restart of the daemon
nsdBufSpace Percentage of the **pagepool** attribute that is reserved for the network transfer of NSD requests	yes	yes	no	yes	if not immediately, on restart of the daemon
nsdInlineWriteMax Specifies the maximum transaction size that can be sent as embedded data in an NSD-write RPC	yes	yes	no	yes	immediately with **-i**
nsdMaxWorkerThreads Sets the maximum number of NSD threads on an NSD server that concurrently transfers data with NSD clients	no	no	no	yes	on restart of the daemon
nsdMinWorkerThreads Used to increase the NSD server performance by providing a large number of dedicated threads for NSD service	no	no	no	yes	on restart of the daemon
nsdMultiQueue Sets the number of queues	yes	yes	no	yes	immediately with **-i**

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
nsdRAIDBufferPoolSizePct Percentage of the page pool that is used for the IBM Spectrum Scale RAID vdisk buffer pool	yes	yes	no	yes	if not immediately, on restart of the daemon
nsdRAIDTracks Number of tracks in the IBM Spectrum Scale RAID buffer pool	yes	yes	no	yes	if not immediately, on restart of the daemon
nsdServerWaitTimeForMount Number of seconds to wait for an NSD server to come up	yes	yes	no	yes	if not immediately, on restart of the daemon
nsdServerWaitTimeWindowOnMount Time window to determine if quorum is to be considered *recently formed*	yes	yes	no	yes	if not immediately, on restart of the daemon
numaMemoryInterleave	no	no	no	yes	on restart of the daemon
pagepool Size of buffer cache on each node	yes	yes	no	yes	if not immediately, on restart of the daemon
pagepoolMaxPhysMemPct Percentage of physical memory that can be assigned to the page pool	no	no	no	yes	on restart of the daemon
pitWorkerThreadsPerNode Maximum number of threads to be involved in parallel processing on each node serving as a Parallel Inode Traversal (PIT) worker	yes	yes	no	yes	immediately with **-i** or **-I**
prefetchPct Acts as a guideline to limit the page pool space that is to be used for prefetch and write-behind buffers for active sequential streams	no	no	no	yes	on restart of the daemon
prefetchThreads Maximum number of threads dedicated to prefetching data	no	no	no	yes	on restart of the daemon
readReplicaPolicy The disk read replica policy	yes	yes	no	yes	immediately with -i
release=LATEST Complete the migration to a new release	yes	no	no	no	if not immediately, on restart of the daemon

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
restripeOnDiskFailure Specifies whether GPFS will attempt to automatically recover from certain common disk failure situations.	yes	no	no	yes	immediately with **-i**
rpcPerfNumberDayIntervals Number of days that aggregated RPC data is saved	no	no	no	yes	on restart of the daemon
rpcPerfNumberHourIntervals Number of hours that aggregated RPC data is saved	no	no	no	yes	on restart of the daemon
rpcPerfNumberMinuteIntervals Number of minutes that aggregated RPC data is saved	no	no	no	yes	on restart of the daemon
rpcPerfNumberSecondIntervals Number of seconds that aggregated RPC data is saved	no	no	no	yes	on restart of the daemon
rpcPerfRawExecBufferSize The buffer size of the raw RPC execution times	no	no	no	yes	on restart of the daemon
rpcPerfRawStatBufferSize The buffer size of the raw RPC statistics	no	no	no	yes	on restart of the daemon
seqDiscardThreshold Detects a sequential read or write access pattern and specifies what has to be done with the page pool buffer after it is consumed or flushed by write-behind threads.	yes	yes	no	yes	immediately with **-i**
sidAutoMapRangeLength Controls the length of the reserved range for Windows SID to UNIX ID mapping	yes	yes	no	no	if not immediately, on restart of the daemon
sidAutoMapRangeStart Specifies the start of the reserved range for Windows SID to UNIX ID mapping	no	no	no	no	on restart of the daemon
syncbuffsperiteration Used to expedite buffer flush and the rename operations done by MapReduce jobs.	yes	yes	no	yes	immediately with **-i**
systemLogLevel Filters messages sent to the system log on Linux	yes	yes	no	yes	if not immediately, on restart of the daemon

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
subnets List of subnets to be used for most efficient daemon-to-daemon communication	no	no	no	yes	on restart of the daemon
sudoUser The default admin user name for logging on to nodes during the processing of an administration command. The GPFS daemon uses this user name only when sudo wrappers are enabled in the cluster and a program running at the root level invokes an administration command directly, without calling the sudo program.	yes	no	no	no	immediately
tiebreakerDisks (CCR repository) List of tiebreaker disks (NSDs)	no	no	no **Note:** If tiebreaker disks are part of the file system, GPFS must be up.	no	immediately
tiebreakerDisks (server-based repository) List of tiebreaker disks (NSDs)	no	no	yes	no	on restart of the daemon
uidDomain The UID domain name for the cluster.	no	no	yes	no	on restart of the daemon
unmountOnDiskFail Unmount the file system on a disk failure	yes	yes	no	yes	if not immediately, on restart of the daemon
usePersistentReserve Enables or disables persistent reserve (PR) on the disks	no	no	yes	no	on restart of the daemon
verbsPorts Specifies InfiniBand device names, port numbers, and IP subnets.	no	no	no	yes	on restart of the daemon
verbsRdma Enables or disables InfiniBand RDMA using the Verbs API.	no	no	no	yes	on restart of the daemon
verbsRdmaCm Enables or disables InfiniBand RDMA_CM using the RDMA_CM API.	no	no	no	yes	on restart of the daemon

Table 4. Configuration attributes on the **mmchconfig** *command (continued)*

Attribute name and Description	-i option allowed	-I option allowed	GPFS must be stopped on all nodes	List of *NodeName*s allowed	Change takes effect
verbsRdmaRoCEToS Specifies the Type of Service (ToS) value for clusters using RDMA over Converged Ethernet (RoCE).	yes	yes	no	yes	if not immediately, on restart of the daemon
verbsRdmaSend Enables or disables the use of InfiniBand RDMA rather than TCP for most GPFS daemon-to-daemon communication.	no	no	no	yes	on restart of the daemon
verbsRecvBufferCount Defines the number of RDMA recv buffers created for each RDMA connection that is enabled for RDMA send when **verbsRdmaSend** is enabled.	no	no	no	yes	on restart of the daemon
verbsRecvBufferSize Defines the size, in bytes, of the RDMA send and recv buffers used for RDMA connections that are enabled for RDMA send when **verbsRdmaSend** is enabled.	no	no	no	yes	on restart of the daemon
workerThreads Sets an integrated group of variables that tune file system performance.	no	no	no	yes	on restart of the daemon
worker1Threads Sets the maximum number of concurrent file operations	yes (only when adjusting value down)	yes (only when adjusting value down)	no	yes	on restart of the daemon
writebehindThreshold Specifies the point at which GPFS starts flushing new data out of the page pool for a file that is being written sequentially.	yes	yes	no	yes	immediately with **-i**

Specify the nodes you want to target for change and the attributes with their new values on the **mmchconfig** command. For example, to change the **pagepool** value for each node in the GPFS cluster immediately, enter:

```
mmchconfig pagepool=100M -i
```

The system displays information similar to:

```
mmchconfig: Command successfully completed
mmchconfig: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

For complete usage information, see **mmchconfig command** in *IBM Spectrum Scale: Command and Programming Reference*.

Security mode

The security mode of a cluster determines the level of security that the cluster provides for communications between nodes in the cluster and also for communications between clusters.

There are three security modes:

EMPTY
> The receiving node and the sending node do not authenticate each other, do not encrypt transmitted data, and do not check the integrity of transmitted data.

AUTHONLY
> The sending and receiving nodes authenticate each other with a TLS handshake and then close the TLS connection. Communication continues in the clear. The nodes do not encrypt transmitted data and do not check data integrity.

Cipher To set this mode, you must specify the name of a supported cipher, such as `AES128-GCM-SHA256`. The sending and receiving nodes authenticate each other with a TLS handshake. A TLS connection is established. The transmitted data is encrypted with the specified cipher and is checked for data integrity.

> To find a list of supported ciphers, choose one of the following methods:
>
> * See the frequently answered questions (FAQs) in IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
> * Enter the following command at the command line:
>
>   `mmauth show ciphers`
>
> For FIPS 140-2 considerations, see the *Encryption* topic in the *IBM Spectrum Scale: Administration Guide*.

For both the **AUTHONLY** mode and the cipher mode, the cluster automatically generates a public/private key pair when the mode is set. However, for communication between clusters, the system administrators are still responsible for exchanging public keys.

In IBM Spectrum Scale V4.2 or later, the default security mode is **AUTHONLY**. The **mmcrcluster** command sets the mode when it creates the cluster. You can display the security mode by running the following command:

`mmlsconfig cipherlist`

You can change the security mode with the following command:

`mmchconfig cipherlist=security_mode`

If you are changing the security mode from **EMPTY** to another mode, you can do so without stopping the GPFS daemon. However, if you are changing the security mode from another mode to **EMPTY**, you must stop the GPFS daemon on all the nodes in the cluster. Change the security mode to **EMPTY** and then restart the GPFS daemon.

The default security mode is **EMPTY** in IBM Spectrum Scale V4.1 or earlier and is **AUTHONLY** in IBM Spectrum Scale V4.2 or later. If you migrate a cluster from IBM Spectrum Scale V4.1 to V4.2 or later by running `mmchconfig release=LATEST`, the command checks the security mode. If the mode is **EMPTY**, the command fails with an error message. You then can do either of two actions:

* Change the security mode to a valid value other than **EMPTY**, such as **AUTHONLY**, and rerun the `mmchconfig release=LATEST` command. Or,
* Leave the security mode set to **EMPTY** and re-run the `mmchconfig release=LATEST` command with the option `--accept-empty-cipherlist-security`.

For more information, see *Completing the migration to a new level of IBM Spectrum Scale* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Configuring the security mode to a setting other than **EMPTY** (that is, either **AUTHONLY** or a supported cipher) requires the use of the GSKit toolkit for encryption and authentication. As such, the **gpfs.gskit** package, which is available on all Editions, should be installed.

Running IBM Spectrum Scale commands without remote root login

With sudo wrapper scripts you can avoid configuring nodes to allow remote root login.

Every administration node in the IBM Spectrum Scale cluster must be able to run administration commands on any other node in the cluster. Each administration node must be able to do so without the use of a password and without producing any extraneous messages. Also, most of the IBM Spectrum Scale administration commands must run at the root level. One solution to meet these requirements is to configure each node to permit general remote login to its root user ID. However, there are secure solutions available that do not require root-level login.

You can use a **sudo** program, or a sudo-like framework to enable a user login, which is not at the root-level. With sudo wrapper, you can launch IBM Spectrum Scale administration commands with a sudo wrapper script. This script uses **ssh** to log in to the remote node using a non-root ID, and then uses sudo on the remote node to run the commands with root-level privileges. The root user on an administration node still needs to be able to log in to all nodes in the cluster as the non-root ID, without being prompted for a password.

Note:
- Sudo wrappers are not supported on clusters where one or more of the nodes is running the Windows operating system.
- Sudo wrappers are not supported with clustered NFS (cNFS).
- Sudo wrappers are not supported with Cluster Export Services (CES).
- Sudo wrappers are not supported with file audit logging.
- The installation toolkit is not supported in a sudo wrapper environment.
- Call home is not supported in a sudo wrapper environment.

To use sudo wrappers, complete the tasks in the following topics:

Configuring sudo

The system administrator must configure sudo by modifying the `sudoers` file. IBM Spectrum Scale installs a sample of the modified `sudoers` file as `/usr/lpp/mmfs/samples/sudoers.sample`.

Do the following steps before you begin to configure sudo:
1. Create a user and group to run administration commands.

   Note: The examples in this section have the user name `gpfsadmin` and the group `gpfs`.
2. Allow the root user from an administration node to run commands on all nodes including the current node with user ID `gpfsadmin` without being prompted for a password. For example, the root user must be able to issue a command like the following one without being prompted for a password:

   `# ssh c6f2bc4n8 -l gpfsadmin /bin/whoami gpfsadmin`
3. Install the sudo program. Sudo is free open-source software that is distributed under a license.

Do the following steps on each node in the cluster:
1. Open the `/etc/sudoers` file with a text editor. The sudo installation includes the *visudo* editor, which checks the syntax of the file before closing.

2. Add the following commands to the file. **Important:** Enter each command on a single line:

```
# Preserve GPFS environment variables:
Defaults env_keep += "MMMODE environmentType GPFS_rshPath GPFS_rcpPath mmScriptTrace GPFSCMDPORTRANGE GPFS_CIM_MSG_FORMAT"

# Allow members of the gpfs group to run all commands but only selected commands without a password:
%gpfs ALL=(ALL) PASSWD: ALL, NOPASSWD: /usr/lpp/mmfs/bin/mmremote, /usr/bin/scp, /bin/echo, /usr/lpp/mmfs/bin/mmsdrrestore

# Disable requiretty for group gpfs:
Defaults:%gpfs !requiretty
```

The first line preserves the environment variables that the IBM Spectrum Scale administration commands need to run. The second line allows the users in the `gpfs` group to run administration commands without being prompted for a password. The third line disables `requiretty`. When this flag is enabled, sudo blocks the commands that do not originate from a TTY session.

3. Perform the following steps to verify that the `sshwrap` and `scpwrap` scripts work correctly.

   a. `sshwrap` is an IBM Spectrum Scale sudo wrapper script for the remote shell command that is installed with IBM Spectrum Scale. To verify that it works correctly, run the following command as the *gpfsadmin* user:

   ```
   sudo /usr/lpp/mmfs/bin/mmcommon test sshwrap nodeName
   [sudo] password for gpfsadmin:
   mmcommon test sshwrap: Command successfully completed
   ```

   Note: Here `nodeName` is the name of an IBM Spectrum Scale node in the cluster.

   b. `scpwrap` is an IBM Spectrum Scale sudo wrapper script for the remote file copy command that is installed with IBM Spectrum Scale. To verify that it works correctly, run the following command as the *gpfsadmin* user:

   ```
   sudo /usr/lpp/mmfs/bin/mmcommon test scpwrap nodeName
   mmcommon test scpwrap: Command successfully completed
   ```

   Note: Here `nodeName` is the name of an IBM Spectrum Scale node in the cluster.

Sudo is now configured to run administration commands without remote root login.

Configuring the cluster to use sudo wrapper scripts

The system administrator must configure the IBM Spectrum Scale cluster to call the sudo wrapper scripts `sshwrap` and `scpwrap` to run IBM Spectrum Scale administration commands. To configure the cluster, run either the **mmcrcluster** command or the **mmchcluster** command with the **--use-sudo-wrapper** option.

Perform the following steps to configure a new cluster or an existing cluster to call the sudo wrapper scripts:

- To configure a new cluster to call the sudo wrapper scripts, use these steps.

  1. Log in with the user ID. This example uses `gpfsadmin` as the user ID.

  2. Issue the **mmcrcluster** command with the **--use-sudo-wrapper** option as shown in the following example:

  ```
  $ sudo /usr/lpp/mmfs/bin/mmcrcluster --use-sudo-wrapper -N c13c1apv7:quorum,c13c1apv8
  mcrcluster: Performing preliminary node verification ...
  mmcrcluster: Processing quorum and other critical nodes ...
  mmcrcluster: Processing the rest of the nodes ...
  mmcrcluster: Finalizing the cluster data structures ...
  mmcrcluster: Command successfully completed mmcrcluster:
  Warning: Not all nodes have proper GPFS license designations.
  Use the mmchlicense command to designate licenses as needed.
  mmcrcluster: Propagating the cluster configuration data to all
  affected nodes. This is an asynchronous process
  ```

  3. To verify that the cluster is using sudo wrappers, issue the **mmlscluster** command as shown in the following example:

```
gpfsadmin@c13c1apv7 admin]$mmlscluster
GPFS cluster information
========================
GPFS cluster name: c13c1apv7.gpfs.net
GPFS cluster id: 12275146245716580740
GPFS UID domain: c13c1apv7.gpfs.net
Remote shell command: /usr/lpp/mmfs/bin/sshwrap
Remote file copy command: sudo wrapper in use
Repository type: CCR
Node Daemon node name IP address Admin node name Designation
-----------------------------------------------------------------
1 c13c1apv7.gpfs.net 192.168.148.117 c13c1apv7.gpfs.net quorum
2 c13c1apv8.gpfs.net 192.168.148.118 c13c1apv8.gpfs.net
```

- To configure an existing cluster to call the sudo wrapper scripts, use these steps.

   1. Log in with the user ID. This example uses `gpfsadmin` as the user ID.
   2. Issue the **mmchcluster** command with the **--use-sudo-wrapper** option to start using the sudo wrappers:

      `sudo /usr/lpp/mmfs/bin/mmchcluster --use-sudo-wrapper`

   3. To verify that cluster is using sudo wrappers, issue the **mmlscluster** command with no parameters. If sudo wrapper is configured properly, the output must contain the following two lines.

      ```
      Remote shell command: sudo wrapper in use
      Remote file copy command: sudo wrapper in use
      ```

Configuring IBM Spectrum Scale GUI to use sudo wrapper

The GUI can be configured to run on a cluster where remote root access is disabled and sudo wrappers are used. On such a cluster, the GUI process still runs as root but it issues ssh to other nodes using a user name for which sudo wrappers were configured.

Make the following configuration changes to use the IBM Spectrum Scale management GUI on a cluster where sudo wrappers are used:

1. Issue the `mmchconfig sudoUser=gpfsadmin` command to configure the user name.
2. Issue the `systemctl restart gpfsgui` command to restart the GUI.

Passwordless ssh is set up between the root user on the node where the GUI is running on all the remote nodes in the cluster. The ssh calls are equivalent to *ssh gpfsadmin@destination-node*. Therefore, it is not necessary to set up passwordless ssh between *gpfsadmin* users on any two nodes. The root user of the node where the GUI is running can do passwordless ssh to any other node using the *gpfsadmin* user login. So, unidirectional access from the GUI node to the remote nodes as *gpfsadmin* user is enough.

Note: If sudo wrappers are enabled on the cluster but GUI is not configured for it, the system raises an event.

Configuring a cluster to stop using sudo wrapper scripts

Follow these directions to stop using sudo wrapper scripts in the IBM Spectrum Scale cluster.

To stop using sudo wrappers, run the **mmchcluster** command with the **--nouse-sudo-wrapper** option as shown in the following example:

`$sudo /usr/lpp/mmfs/bin/mmchcluster --nouse-sudo-wrapper`

The cluster stops calling the sudo wrapper scripts to run the remote administration commands.

Root-level processes that call administration commands directly

With the `sudoUser` attribute, you can enable root-level background processes to call administration commands directly while sudo wrappers are enabled.

When sudo wrappers are enabled and a root-level background process calls an administration command directly rather than through **sudo**, the administration command typically fails. Examples of such a root-level process are the **cron** program and IBM Spectrum Scale callback programs. Such processes call administration commands directly even when sudo wrappers are enabled.

In the failing scenario, the GPFS daemon that processes the administration command encounters a login error when it tries to run an internal command on another node as the root user. When sudo wrappers are enabled, nodes typically do not allow root-level logins by other nodes. (That is the advantage of having sudo wrappers.) When the root-level login fails, the GPFS daemon that is processing the administration command cannot complete the command and returns an error.

To avoid this problem, you can set the **sudoUser** attribute to a non-root admin user ID that can log in to any node in the cluster without being prompted for a password. You can specify the same admin user ID that you used to configure **sudo**. For more information on the admin user ID, see "Configuring sudo" on page 17.

You can set the **sudoUser** attribute in the following commands in the *IBM Spectrum Scale: Command and Programming Reference*: *mmchconfig command* (the **sudoUser** attribute), *mmcrcluster command* (the **--sudo-user** parameter), or *mmcrcluster command* (the **--sudo-user** parameter).

Node quorum considerations

A node quorum is the minimum number of nodes that must be running in order for the daemon to start. Node quorum is the default quorum algorithm for GPFS.

For more information on node quorum, see the section on *Quorum*, in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

Node quorum with tiebreaker considerations

Node quorum with tiebreaker disks allows you to run with as few as one quorum node available as long as you have access to a majority of the quorum disks. Enabling node quorum with tiebreaker disks starts by designating one or more nodes as quorum nodes. Then one to three disks are defined as tiebreaker disks using the **tiebreakerDisks** parameter on the **mmchconfig** command. You can designate any disk to be a tiebreaker. When utilizing node quorum with tiebreaker disks, there are specific rules for cluster nodes and for tiebreaker disks.

For more information on node quorum with tiebreaker, see the section on *Quorum* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

When using node quorum with tiebreaker, define one, two, or three disks to be used as tiebreaker disks when any quorum node is down. Issue this command:

```
mmchconfig tiebreakerDisks="nsdName;nsdName;nsdName"
```

Consider these points:
- You are not permitted to change a GPFS cluster configuration to use node quorum with tiebreaker if there are more than eight existing quorum nodes.
- You can have a maximum of three tiebreaker disks.
- The disks must be directly attached to all quorum nodes.
- When adding tiebreaker disks:
  - If the tiebreaker disks are part of a file system, GPFS should be up and running.
  - If the tiebreaker disks are not part of a file system, GPFS can be either running or shut down.
- When using the traditional server-based (non-CCR) configuration repository, the GPFS daemons must be down on all nodes in the cluster when running **mmchconfig tiebreakerDisks**.

If you are using node quorum with tiebreaker and want to change to using node quorum, issue this command:

```
mmchconfig tiebreakerDisks=DEFAULT
```

Displaying and changing the file system manager node

In general, GPFS performs the same functions on all nodes. There are also cases where one node provides a more global function that affects the operation of multiple nodes. For example, each file system is assigned a node that functions as a file system manager.

For a more detailed discussion on the role of the file system manager node, see *Special management functions* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The node that is the file system manager can also be used for applications. In some cases involving very large clusters or applications that place a high stress on metadata operations, it may be useful to specify which nodes are used as file system managers. Applications that place a high stress on metadata operations are usually those that involve large numbers of very small files, or that do very fine-grain parallel write-sharing among multiple nodes.

You can display the file system manager node by issuing the **mmlsmgr** command. You can display the information for an individual file system, a list of file systems, or for all of the file systems in the cluster. For example, to display the file system manager for the file system **fs1**, enter:

```
mmlsmgr fs1
```

The output shows the device name of the file system and the file system manager's node number and name:

```
file system       manager node        [from 19.134.68.69 (k164n05)]
---------------- ------------------
fs1               19.134.68.70 (k164n06)
```

For complete usage information, see `mmlsmgr command` in *IBM Spectrum Scale: Command and Programming Reference*.

You can change the file system manager node for an individual file system by issuing the **mmchmgr** command. For example, to change the file system manager node for the file system **fs1** to **k145n32**, enter:

```
mmchmgr fs1 k145n32
```

The output shows the file system manager's node number and name, in parentheses, as recorded in the GPFS cluster data:

```
GPFS: 6027-628 Sending migrate request to current manager node 19.134.68.69 (k145n30).
GPFS: 6027-629 [N] Node 19.134.68.69 (k145n30) resigned as manager for fs1.
GPFS: 6027-630 [N] Node 19.134.68.70 (k145n32) appointed as manager for fs1.
```

For complete usage information, see `mmchmgr command` in *IBM Spectrum Scale: Command and Programming Reference*.

Determining how long `mmrestripefs` takes to complete

There are several factors that determine how long the **mmrestripefs** command takes to complete.

To determine how long the **mmrestripefs** command takes to complete, consider these points:
1. How much data potentially needs to be moved. You can estimate this using the **df** command.
2. How many GPFS client nodes there are to do the work.
3. How much Network Shared Disk (NSD) server bandwidth is available for I/O operations.

4. If you add new disks to a file system, use the **mmdf** command to determine how much free space is available on each of the new disks.

The restriping of a file system is done by having multiple threads on each node in the cluster work on a subset of files. If the files are large, multiple nodes can participate in restriping it in parallel. So, the more GPFS client nodes that are performing work for the restripe, the faster the **mmrestripefs** command completes. The nodes that should participate in the restripe are specified on the command by using the **-N** parameter. Based on raw I/O rates, you should be able to estimate the length of time for the restripe. However, to account for the overhead of scanning all metadata, that value should be doubled.

Assuming that you have enough nodes to saturate the disk servers, and have to move all of the data, the time to read and write every block of data is roughly:

```
2 * fileSystemSize / averageDiskserverDataRate
```

As an upper bound, due to overhead of scanning all of the metadata, this time should be doubled. If other jobs are loading the NSD servers heavily, this time might increase even more.

Note: There is no particular reason to stop all other jobs while the **mmrestripefs** command is running. The CPU load of the command is minimal on each node and only the files that are being restriped at any moment are locked to maintain data integrity.

Starting and stopping GPFS

You can use the **mmstartup** and **mmshutdown** commands to start and stop GPFS on new or existing clusters.

For new GPFS clusters, see *Steps to establishing and starting your GPFS cluster* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For existing GPFS clusters, before starting GPFS, ensure that you have:
1. Verified the installation of all prerequisite software.
2. Compiled the GPL layer, if Linux is being used.
3. Properly configured and tuned your system for use by GPFS. This should be done prior to starting GPFS.

For details, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Start the daemons on all of the nodes in the cluster by issuing the **mmstartup -a** command:
```
mmstartup  -a
```

The output is similar to this:
```
Tue Aug 24 15:54:56 edt 2004: 6027-1642 mmstartup: Starting GPFS ...
```

Check the messages recorded in **/var/adm/ras/mmfs.log.latest** on one node for verification. Look for messages similar to this:
```
GPFS: 6027-300 [N] mmfsd ready
```

This indicates that quorum has been formed and this node has successfully joined the cluster, and is now ready to mount file systems.

If GPFS does not start, see *GPFS daemon will not come up* in *IBM Spectrum Scale: Problem Determination Guide*.

For complete usage information, see `mmstartup command` in *IBM Spectrum Scale: Command and Programming Reference*.

If it becomes necessary to stop GPFS, you can do so from the command line by issuing the **mmshutdown** command:

```
mmshutdown -a
```

The system displays information similar to:

```
Thu Aug 12 13:10:40 EDT 2004: 6027-1341 mmshutdown: Starting force unmount of GPFS file systems
k164n05.kgn.ibm.com:  forced unmount of /fs1
k164n04.kgn.ibm.com:  forced unmount of /fs1
k164n06.kgn.ibm.com:  forced unmount of /fs1
Thu Aug 12 13:10:45 EDT 2004: 6027-1344 mmshutdown: Shutting down GPFS daemons
k164n04.kgn.ibm.com:  Shutting down!
k164n06.kgn.ibm.com:  Shutting down!
k164n05.kgn.ibm.com:  Shutting down!
k164n04.kgn.ibm.com:  'shutdown' command about to kill process 49682
k164n05.kgn.ibm.com:  'shutdown' command about to kill process 28194
k164n06.kgn.ibm.com:  'shutdown' command about to kill process 30782
Thu Aug 12 13:10:54 EDT 2004: 6027-1345 mmshutdown: Finished
```

For complete usage information, see `mmshutdown command` in *IBM Spectrum Scale: Command and Programming Reference*.

Shutting down an IBM Spectrum Scale cluster

Use the following information to shut down an IBM Spectrum Scale cluster in an emergency situation.

1. Stop the protocol services on all protocol nodes in the cluster using the `mmces service stop` command. For example:

   ```
   mmces service stop nfs -a
   mmces service stop smb -a
   mmces service stop obj -a
   ```

2. Unmount all file systems, except the CES shared root file system, on all nodes in the cluster using the `mmumount` command.
3. Stop GPFS daemons on all protocol nodes in the cluster using the `mmshutdown -N cesNodes` command.
4. Unmount all file systems on all nodes in the cluster using the `mmumount all -a` command.
5. Stop GPFS daemons on all nodes in the cluster using the `mmshutdown -a` command.

After performing these steps, depending on your operating system, shut down your servers accordingly.

Before shutting down and powering up your servers, consider the following:

- You must shut down NSD servers before the storage subsystem. While powering up, the storage subsystem must be online before NSD servers are up so that LUNs are visible to them.
- In a power-on scenario, verify that all network and storage subsystems are fully operational before bringing up any IBM Spectrum Scale nodes.
- On the Power® platform, you must shut down operating systems for LPARs first and then power off servers using Hardware Management Console (HMC). HMC must be the last to be shut down and the first to be powered up.
- It is preferable to shut down your Ethernet and InfiniBand switches using the management console instead of powering them off. In any case, network infrastructure such as switches or extenders must be powered off last.
- After starting up again, verify that functions such as AFM and policies are operational. You might need to manually restart some functions.
- There are a number other GPFS functions that could be interrupted by a shutdown. Ensure that you understand what else might need to be verified depending on your environment.

Chapter 2. Configuring the CES and protocol configuration

After GPFS is configured, Cluster Export Services (CES) and its protocols can be configured, administered, or removed from the system.

Some of the CES and protocol configuration steps might have been completed already through the IBM Spectrum Scale installer. To verify, see the information about the IBM Spectrum Scale installer and protocol configuration in the topic *spectrumscale command* in the *IBM Spectrum Scale: Command and Programming Reference* guide.

A manual or a minimal installation of CES involves configuration and administrative tasks.

Configuring Cluster Export Services

If you have not configured Cluster Export Services (CES) through the installer, you must configure CES now.

For more information on the CES features, see Chapter 31, "Implementing Cluster Export Services," on page 459.

Setting up Cluster Export Services shared root file system

If you have not set up a shared root file system through the installer, create one for Cluster Export Services (CES).

The CES shared root (cesSharedRoot) is needed for storing CES shared configuration data, for protocol recovery, and for other protocol-specific purposes. It is part of the cluster export configuration and is shared between the protocols. Every CES node requires access to the path configured as shared root.

The **mmchconfig** command is used to configure this directory as part of setting up a CES cluster.

The cesSharedRoot cannot be changed while any CES nodes are up and running. You need to bring down all CES nodes if you want to modify the shared root configuration.

The cesSharedRoot is monitored by the **mmsysmonitor**. If the shared root is not available, the CES node list (**mmces node list**) will show "no-shared-root" and a failover is triggered.

The cesSharedRoot cannot be unmounted when the CES cluster is up and running. You need to bring all CES nodes down if you want to unmount cesSharedRoot (for example, for doing service action like fsck).

To list the current cesSharedRoot, run

```
mmlsconfig cesSharedRoot
cesSharedRoot /gpfs/gpfs-ces/
```

The recommendation for CES shared root is a dedicated file system (but this is not enforced). It can also be a part (path) of an existing GPFS file system. A dedicated file system can be created with the **mmcrfs** command. In any case, CES shared root must reside on GPFS and must be available when it is configured through **mmchconfig**.

If not already done through the installer, it is recommended that you create a file system for the CES. Some protocol services share information through a cluster-wide file system. It is recommended to use a separate file system for this purpose.

Note: The recommended size for CES shared root file system is greater than or equal to 4GB.

To set up CES, change the configuration to use the new file system:

```
mmchconfig cesSharedRoot=/gpfs/fs0
```

Note: Once GPFS starts back up, by virtue of the fact that the cesSharedRoot is now defined, then CES can be enabled on the cluster.

Configuring Cluster Export Services nodes

If you have not configured Cluster Export Services (CES) nodes through the installer, you must configure them before you configure any protocols.

If not already done during the installation, this must be done before configuring any protocols. Nodes that should participate in the handling of protocol exports need to be configured as CES nodes.

Note: You can have a maximum of 16 nodes in a CES cluster.

For each of the nodes that should handle protocol exports, run:

```
mmchnode -N nodename --ces-enable
```

After configuring all nodes, verify that the list of CES nodes is complete:

```
mmces node list
```

CES nodes may be assigned to CES groups. A CES group is identified by a group name consisting of lowercase alphanumeric characters. CES groups may be used to manage CES node and address assignments.

Nodes may be assigned to groups by issuing the following command:

```
mmchnode --ces-group group1 -N node
```

A node may be assigned to multiple groups by issuing the following command:

```
mmchnode --ces-group group1,group2,group3 -N node1,node2
```

The group assignment may also be specified when the node is enabled for CES by issuing the following command:

```
mmchnode --ces-enable --ces-group group1,group2 -N node
```

The node may be removed from a group at any time by issuing the following command:

```
mmchnode --noces-group group1 -N node
```

For more information, see *mmchnode command* in *IBM Spectrum Scale: Command and Programming Reference*.

Configuring CES protocol service IP addresses

Protocol services are made available through Cluster Export Services (CES) protocol service IP addresses. These addresses are separate from the IP addresses that are used internally by the cluster.

Each CES protocol service IP address is assigned initially to one CES node, either explicitly as specified by the **mmces address add** command, or by the system, but they can be moved later either manually or automatically in response to certain events.

```
mmces address add --ces-node Node1 --ces-ip 192.168.6.6
```

After adding all desired CES protocol service IP addresses, verify the configuration:

```
mmces address list
```

Use **mmces address add --ces-ip 192.168.6.6** to add an IP address to the CES IP address pool. The IP address will be assigned to a CES node according to the CES "Address distribution policy".

CES addresses can be assigned to CES groups. A CES group is identified by a group name consisting of alphanumeric characters which are case-sensitive. Addresses can be assigned to a group when they are defined by issuing the following command:

```
mmces address add --ces-ip 192.168.6.6 --ces-group group1
```

The group assignment can be changed by issuing the following command:

```
mmces address change --ces-ip 192.168.6.6 --ces-group group2
```

The group assignment can be removed by issuing the following command:

```
mmces address change --ces-ip 192.168.6.6 --remove-group
```

A CES address that is associated with a group must be assigned only to a node that is also associated with the same group. A node can belong to multiple groups while an address cannot.

As an example, consider a configuration with three nodes. All three nodes can host addresses on subnet A, and two of the nodes can host addresses on subnet B. The nodes must have existing non-CES IP address of the same subnet configured on the interfaces intended to be used for the CES IPs. Also four addresses are defined, two on each subnet.

Node1: groups=subnetA,subnetB

Node2: groups=subnetA,subnetB

Node3: groups=subnetA

Address1: subnetA

Address2: subnetA

Address3: subnetB

Address4: subnetB

In this example, Address1 and Address2 can be assigned to any of the three nodes, but Address3 and Address4 can be assigned to only Node1 or Node2.

If an address is assigned to a group for which there are no healthy nodes, the address will remain unassigned until a node in the same group becomes available.

Addresses without a group assignment can be assigned to any node. Therefore, it is necessary to use a group for each subnet when multiple subnets exist.

For more information, see *mmces command* in *IBM Spectrum Scale: Command and Programming Reference*.

CES IP aliasing to network adapters on protocol nodes

Cluster Export Services (CES) is a functionality in IBM Spectrum Scale that enables NFS, SMB, and Object protocols. Irrespective of which protocols you choose, all are accessible through a floating pool of IP addresses called CES IPs. This pool of CES IPs is considered floating because each IP can move independently among all protocol nodes. In the event of a protocol node failure, accessibility to all protocols is maintained as the CES IPs automatically move from the failed protocol node to a healthy protocol node. Use this information to understand how CES IPs are assigned and how they are aliased to adapters with or without VLAN tagging.

Virtual LANs (VLANs) are often associated with secure networks because they provide a means of separating network devices into independent networks. Although the physical network infrastructure is shared, unicast, multicast, and broadcast traffic from a network device in a VLAN is restricted to other devices within that same VLAN.

How are CES IPs assigned

CES IPs are automatically assigned and aliased to existing network adapters on protocol nodes during startup. The following example shows aliased CES IPs in a flat network environment or a single VLAN environment in which the switch ports are set to Access mode and thus, do not need VLAN tagging.
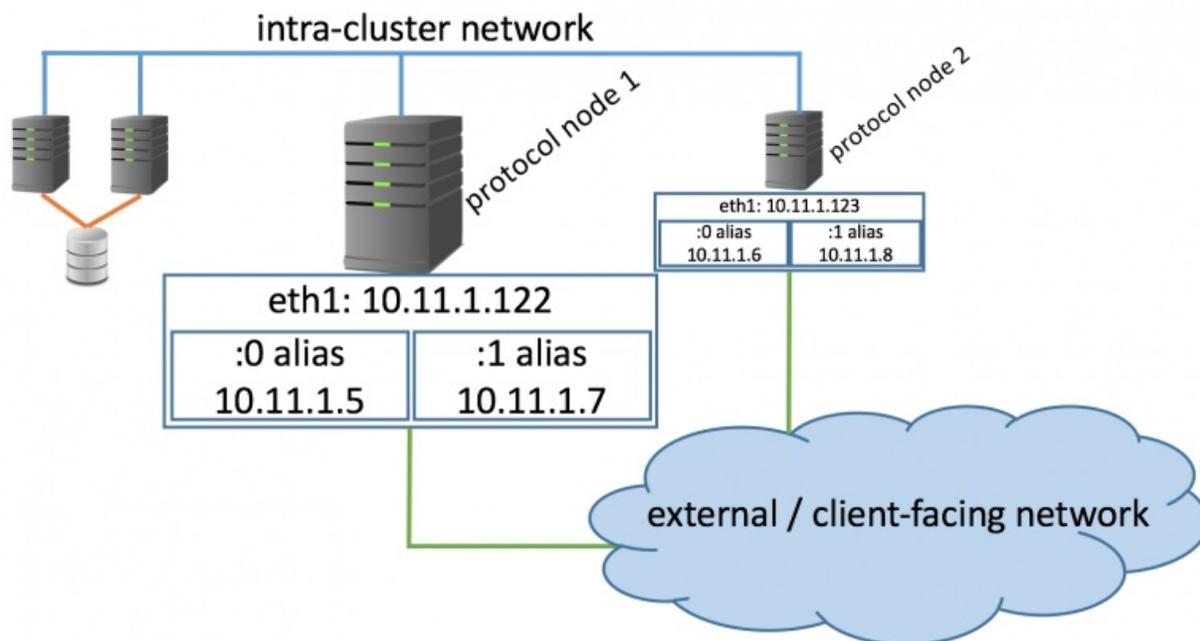
Example of aliased CES IPs using the `ip addr` command

```
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP qlen 1000
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
inet 10.11.1.122/24 brd 10.11.1.255 scope global eth1
  valid_lft forever preferred_lft forever
inet 10.11.1.5/24 brd 10.11.1.255 scope global secondary eth1:0
  valid_lft forever preferred_lft forever
inet 10.11.1.7/24 brd 10.11.1.255 scope global secondary eth1:1
  valid_lft forever preferred_lft forever
```

Example of preexisting routes

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref   Use  Iface
default         gateway         0.0.0.0         UG    100    0     0    eth1
10.11.1.0       0.0.0.0         255.255.255.0   U     100    0     0    eth1
172.31.128.0    0.0.0.0         255.255.128.0   U     300    0     0    data0
```

In the preceding example, `eth1` preexists with an established route and IP: 10.11.1.122. This is manually assigned and must be accessible prior to any CES configuration. Once CES services are active, CES IPs are then automatically aliased to this base adapter, thus creating `eth1:0` and `eth1:1`. The floating CES IPs assigned to the aliases are 10.11.1.5 and 10.11.1.7. Both CES IPs are allowed to move to other nodes in case of a failure. This automatic movement combined with the ability to manually move CES IPs, might

cause a variance in the number of aliases and CES IPs among protocol nodes. The `data0` interface illustrates how a network used for GPFS intra-cluster connectivity between nodes can be separate from the adapter used for CES IPs.

Example distribution of CES IPs among two protocol nodes after enablement of protocols

```
mmces address list
Address         Node                    Group      Attribute
------------------------------------------------------------------------
10.11.1.5       protocol-node-1         none       none
10.11.1.6       protocol-node-2         none       object_database_node,object_singleton_node
10.11.1.7       protocol-node-1         none       none
10.11.1.8       protocol-node-2         none       none
```

CES IPs and VLAN tags

A network switch port can be considered a trunk port if it gives access to multiple VLANs. When this occurs, it is necessary for a VLAN tag to be added to each frame. This VLAN tag is an identification allowing switches to contain traffic within specific networks. If multiple networks must access data from IBM Spectrum Scale protocol nodes, then one possible option is to configure trunk ports on the switch directly connected to the IBM Spectrum Scale protocol nodes. Once a trunk port exists, VLAN tags are necessary on the connected network adapters. Note that CES IPs are automatically assigned and aliased to existing network adapters on protocol nodes during startup. Due to this, the existence of VLAN tags requires a preexisting network adapter with an established route and IP so that CES IPs can alias to it.
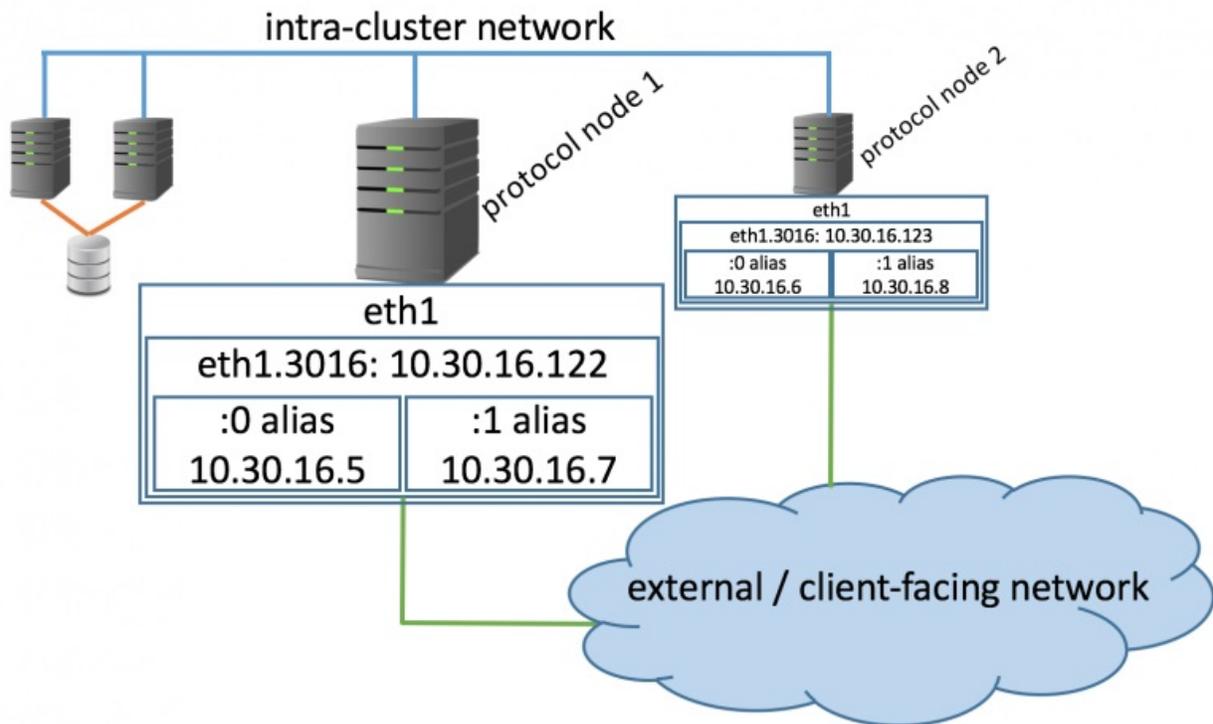
Example of aliased CES IPs using the `ip addr` command (with VLAN tag)

```
eth1: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
    valid_lft forever preferred_lft forever

eth1.3016: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP qlen 1000
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
inet 10.30.16.122/24 brd 10.30.16.255 scope global eth1.3016
```

Chapter 2. Configuring the CES and protocol configuration **29**

```
      valid_lft forever preferred_lft forever
inet 10.30.16.5/24 brd 10.30.16.255 scope global secondary eth1.3016:0
   valid_lft forever preferred_lft forever
inet 10.30.16.7/24 brd 10.30.16.255 scope global secondary eth1.3016:1
   valid_lft forever preferred_lft forever
```

Example of pre-existing routes (with VLAN tag)

```
Kernel IP routing table
Destination     Gateway     Genmask         Flags Metric Ref Use   Iface
default         gateway     0.0.0.0         UG    100    0   0     eth1.3016
10.30.16.0      0.0.0.0     255.255.255.0   U     100    0   0     eth1.3016
172.31.128.0    0.0.0.0     255.255.128.0   U     300    0   0     data0
```

As in the no VLAN tag example, an existing network adapter must be present so that CES\ IPs can alias to it. Note the non-VLAN base adapter eth1, has no IPs assigned. In this example, the preexisting network adapter with an established route and IP is `eth1.3016`. The IP for `eth1.3016` is 10.30.16.122 and the VLAN tag is 3016. This preexisting IP can be used for network verification prior to CES IP configuration by pinging it from external to the cluster or pinging it from other protocol nodes. It is a good practice to make sure that all protocol node base adapter IPs are accessible before enabling protocols. The `data0` interface shows how a network used for GPFS intra-cluster connectivity between nodes can be separate from the adapter used for CES IPs.

Example distribution of CES IPs among two protocol nodes after enablement of protocols (with VLAN tag)

```
mmces address list
Address         Node                       Group      Attribute
---------------------------------------------------------------------
10.30.16.5      protocol-node-1            none       none
10.30.16.6      protocol-node-2            none       object_database_node,object_singleton_node
10.30.16.7      protocol-node-1            none       none
10.30.16.8      protocol-node-2            none       none
```

CES IPs and multiple VLAN tags

The following diagram shows a node with two network adapters devoted to CES protocols: `eth1` and `eth2`. The `eth1` interface has two VLANs associated: 3016 and 3017. The `eth2` interface has one VLAN associated: 80.
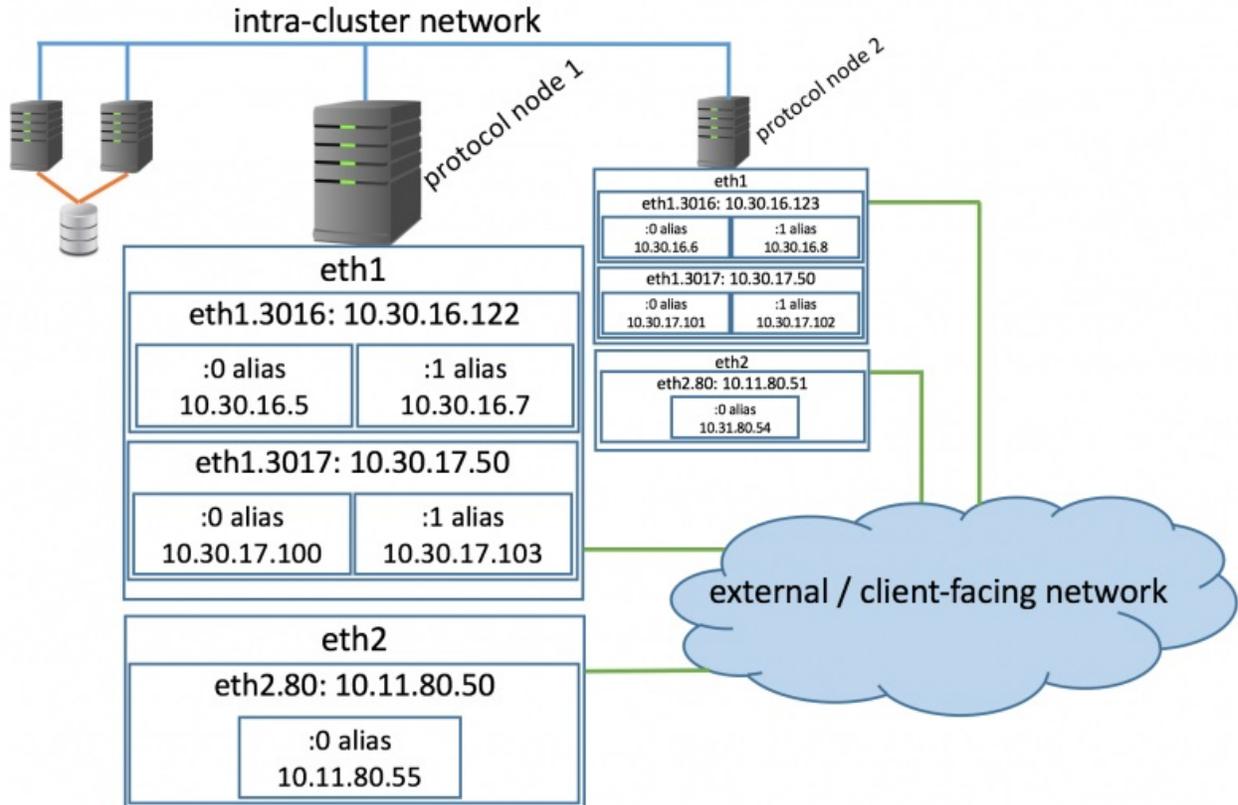
Example of aliased CES IPs using the `ip addr` command (with multiple VLAN tag)

```
eth1: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
    valid_lft forever preferred_lft forever

eth1.3016: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP qlen 1000
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
inet 10.30.16.122/24 brd 10.30.16.255 scope global eth1.3016
    valid_lft forever preferred_lft forever
inet 10.30.16.5/24 brd 10.30.16.255 scope global secondary eth1.3016:0
    valid_lft forever preferred_lft forever
inet 10.30.16.7/24 brd 10.30.16.255 scope global secondary eth1.3016:1
    valid_lft forever preferred_lft forever

eth1.3017: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP qlen 1000
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
inet 10.30.17.50/24 brd 10.30.17.255 scope global eth1.3017
    valid_lft forever preferred_lft forever
inet 10.30.17.100/24 brd 10.30.16.255 scope global secondary eth1.3017:0
    valid_lft forever preferred_lft forever
inet 10.30.17.103/24 brd 10.30.16.255 scope global secondary eth1.3017:1
    valid_lft forever preferred_lft forever

eth2: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 qdisc noqueue state UNKNOWN
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
    valid_lft forever preferred_lft forever

eth2.80: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP qlen 1000
  link/ether 00:50:56:83:16:e5 brd ff:ff:ff:ff:ff:ff
```

```
inet 10.11.80.50/24 brd 10.11.80.255 scope global eth1.80
   valid_lft forever preferred_lft forever
inet 10.11.80.55/24 brd 10.11.80.255 scope global secondary eth1.80:0
   valid_lft forever preferred_lft forever
```

Example of preexisting routes (with multiple VLAN tag)

```
Kernel IP routing table
Destination     Gateway       Genmask          Flags Metric Ref   Use  Iface
default         gateway       0.0.0.0          UG    100    0     0    eth1.3016
10.30.16.0      0.0.0.0       255.255.255.0    U     100    0     0    eth1.3016
10.30.17.0      0.0.0.0       255.255.255.0    U     400    0     0    eth1.3017
10.11.80.0      0.0.0.0       255.255.255.0    U     200    0     0    eth2.80
172.31.128.0    0.0.0.0       255.255.128.0    U     300    0     0    data0
```

Example distribution of CES IPs from multiple VLANs among two protocol nodes after enablement of protocols

```
mmces address list
Address         Node                          Group      Attribute
-------------------------------------------------------------------------
10.11.80.54     protocol-node-2               none       none
10.11.80.55     protocol-node-1               none       none
10.30.16.5      protocol-node-1               none       none
10.30.16.6      protocol-node-2               none       none
10.30.16.7      protocol-node-1               none       none
10.30.16.8      protocol-node-2               none       none
10.30.17.100    protocol-node-1               none       none
10.30.17.101    protocol-node-2               none       none
10.30.17.102    protocol-node-2               none       object_database_node,object_singleton_node
10.30.17.103    protocol-node-1               none       none
```

Deploying Cluster Export Services packages on existing IBM Spectrum Scale 4.1.1 and later nodes

Use the following instructions to copy packages on your protocol nodes and to deploy these packages.

1. Copy the required packages to the protocol node from the location where the self-extracting package was extracted.

   By default, installation images are extracted to the target directory `/usr/lpp/mmfs/4.2.1.0`

2. Install packages by issuing the following command: **rpm -ivh** *Package_Name1 Package_Name2 ... Package_NameN*

   For a list of packages applicable for the current IBM Spectrum Scale release, see *Manually installing the software packages on Linux nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

3. Set the server licenses for each CES node by issuing the following command: **mmchlicense server --accept -N ces_node_ips**

   For example:

   ```
   mmchlicense server --accept -N 203.0.113.7,203.0.113.9
   ```

4. Enable CES by issuing the following command: **mmchnode -N ces_nodes --ces-enable**

   For example:

   ```
   mmchnode -N 203.0.113.7,203.0.113.9 --ces-enable
   ```

5. Assign export IPs for each export_IP by issuing this command: **mmces address add --ces-ip export_IP**

Verifying the final CES configurations

After you finish the Cluster Export Services (CES) configuration steps, verify the final configuration.

To verify your configuration, run the following command:

```
mmlscluster --ces
```

For more information, see the `mmces node list` and `mmces address list` options in *mmces command* in *IBM Spectrum Scale: Command and Programming Reference*.

Creating and configuring file systems and filesets for exports

If you have not done so previously, create the file systems and the filesets for the data to be exported through the protocol services. For more information, see *mmcrfs* and *mmcrfileset* in *IBM Spectrum Scale: Command and Programming Reference*.

Creating a fileset through the GPFS GUI

To create a fileset, log on to the IBM Spectrum Scale GUI and select **Files** > **Filesets** > **Create Fileset**.

When the file system is intended for CES export, IBM strongly recommends to configure the file systems to only allow NFSv4 ACLs through the -k nfs4 option for mmcrfs. When using the default configuration profiles (/usr/lpp/mmfs/profiles) that are included with IBM Spectrum Scale, the NFSv4 ACL setting is already set from the profile configuration (see "Authorizing file protocol users" on page 316 for details). Also if quotas should be used, enable the quota usage during the file system creation.

For information on unified file and object access, see *Planning for unified file and object access* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Note: Ensure that all GPFS file systems used to export data via NFS are mounted with the `syncnfs` option in order to prevent clients from running into data integrity issues during failover. It is recommended to use the `mmchfs` command to set the `syncnfs` option as default when mounting the GPFS file system.

For more information on creating protocol data exports, see *File system considerations for the NFS protocol* and *Fileset considerations for creating protocol data exports* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Configuring with the installation toolkit

You can use the configuration options of the installation toolkit to configure GPFS and protocols on an ongoing basis, as an alternative to the other GPFS cluster creation and configuration commands.

For detailed information about using the installation toolkit to configure GPFS and protocols, see the following:
* **spectrumscale command** in *IBM Spectrum Scale: Command and Programming Reference*
* *Installing IBM Spectrum Scale on Linux nodes and deploying protocols* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
* *Using the spectrumscale installation toolkit to perform installation tasks: Explanations and examples* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Deleting a Cluster Export Services node from an IBM Spectrum Scale cluster

Use this information to delete a Cluster Export Services (CES) node from an IBM Spectrum Scale cluster.

1. On a node other than the one you want to delete from the cluster, issue the following command to suspend the node.

   `# mmces node suspend —N <Node_to_Delete>`

2. On the node that you want to delete from the cluster, issue the following commands to stop the CES services.

```
# mmces service stop nfs
# mmces service stop smb
# mmces service stop obj
```

In this example, it is assumed that all three protocols are enabled on the node that you want to delete from the cluster.

3. On a node other than the one you want to delete from the cluster, issue the following command to disable CES on the node.

```
# mmchnode –N <Node_to_Delete> --ces-disable
```

4. On a node other than the one you want to delete from the cluster, issue the following command to shut down GPFS on the node.

```
# mmshutdown –N <Node_to_Delete>
```

5. On a node other than the one you want to delete from the cluster, issue the following command to delete the node from the cluster.

```
# mmdelnode –N <Node_to_Delete>
```

Setting up Cluster Export Services groups in an IBM Spectrum Scale cluster

After IBM Spectrum Scale is successfully installed and deployed, you can set up Cluster Export Services (CES) groups that are specific to nodes and CES IPs on a cluster that is working correctly by using the following information.

1. Set the CES nodes in the cluster to the corresponding groups by issuing the **mmchnode --ces-group** command. For example:

```
mmchnode --ces-group Site1 -N prt001st001
mmchnode --ces-group Site1 -N prt002st001
mmchnode --ces-group Site2 -N prt003st001
mmchnode --ces-group Site2 -N prt004st001
```

Note: CES group names are not case-sensitive.

In the example, protocol nodes prt001st001 and prt002st001 are set to the Site1 CES group, and protocol nodes prt003st001 and prt004st001 are set to the site2 CES group.

2. Assign CES IPs to the corresponding CES groups by issuing the **mmces address change** command. For example:

```
mmces address change --ces-ip 192.0.2.20,192.0.2.21,192.0.2.22,192.0.2.23
--ces-group Site1
mmces address change --ces-ip 192.0.3.20,192.0.3.21,192.0.3.22
--ces-group Site2
```

3. To verify the CES groups your nodes belong to, issue the **mmces node list command**.

The system displays information similar to this:

```
Node Name                                    Node Flags    Node Groups
-----------------------------------------------------------------
10   prt005st001                             none          site2
11   prt006st001                             none          site2
12   prt007st001                             none          site2
13   prt008st001                             none          site2
6    prt001st001                             none          site1
7    prt002st001                             none          site1
8    prt003st001                             none          site1
9    prt004st001                             none          site1
```

4. To verify the groups your CES IPs belong to, issue the **mmces address list command.** The system displays information similar to the following:

```
Address        Node                              Group    Attribute
------------------------------------------------------------------------
10.18.52.30    prt001st001                       site1    object_singleton_node,
```

```
                                                      object_database_node
10.18.52.31    prt002st001                    site1   none
10.18.52.32    prt003st001                    site1   none
10.18.52.33    prt004st001                    site1   none
10.18.60.30    prt005st001                    site2   none
10.18.60.31    prt006st001                    site2   none
10.18.60.32    prt007st001                    site2   none
10.18.60.33    prt008st001                    site2   none
```

Chapter 3. Configuring and tuning your system for GPFS

In addition to configuring your GPFS cluster, you need to configure and tune your system.

For more information, see *GPFS cluster creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Values suggested here reflect evaluations made at the time this documentation was written. For the latest system configuration and tuning settings, see the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) and the IBM Spectrum Scale Wiki (www.ibm.com/developerworks/community/wikis/home/wiki/General Parallel File System (GPFS)).

Additional GPFS and system configuration and tuning considerations include:
1. "General system configuration and tuning considerations"
2. "Linux configuration and tuning considerations" on page 41
3. "AIX configuration and tuning considerations" on page 43

For information on installing and configuring Windows on systems that will be added to a GPFS cluster, see *Configuring Windows* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For more information on using multiple token servers, see "Using multiple token servers" on page 684

General system configuration and tuning considerations

You need to take into account some general system configuration and tuning considerations. This topic points you to the detailed information.

For the latest system configuration settings, see the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Configuration and tuning considerations for all systems include:
1. "Clock synchronization"
2. "GPFS administration security"
3. "Cache usage" on page 38
4. Chapter 4, "Parameters for performance tuning and optimization," on page 45
5. "Access patterns" on page 40
6. "Aggregate network interfaces" on page 40
7. "Swap space" on page 41

Clock synchronization

The clocks of all nodes in the GPFS cluster must be synchronized. If this is not done, NFS access to the data and other GPFS file system operations may be disrupted.

GPFS administration security

Before administering your GPFS file system, make certain that your system has been properly configured for security.

This includes:

- Assigning root authority to perform all GPFS administration tasks except:
  - Tasks with functions limited to listing GPFS operating characteristics.
  - Tasks related to modifying individual user file attributes.
- Establishing the authentication method between nodes in the GPFS cluster.
  - Until you set the authentication method, you cannot issue any GPFS commands.
- Designating a remote communication program for remote shell and remote file copy commands.
  - The default remote communication commands are **scp** and **ssh**. You can designate any other remote commands if they have the same syntax.
  - Regardless of which remote commands have been selected, the nodes that you plan to use for administering GPFS must be able to execute commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Cache usage

GPFS creates a number of cache segments on each node in the cluster. The amount of cache is controlled by three attributes.

These attributes have default values at cluster creation time and may be changed through the **mmchconfig** command:

pagepool
> The GPFS **pagepool** attribute is used to cache user data and file system metadata. The **pagepool** mechanism allows GPFS to implement read as well as write requests asynchronously. Increasing the size of the **pagepool** attribute increases the amount of data or metadata that GPFS can cache without requiring synchronous I/O. The amount of memory available for GPFS on a particular node may be restricted by the operating system and other software running on the node.
>
> The optimal size of the **pagepool** attribute depends on the needs of the application and effective caching of its re-accessed data. For systems where applications access large files, reuse data, benefit from GPFS prefetching of data, or have a random I/O pattern, increasing the value for the **pagepool** attribute may prove beneficial. However, if the value is set too large, GPFS will start with the maximum that the system allows. See the GPFS log for the value it is running at.
>
> To change the size of the **pagepool** attribute to 4 GB:
>
> ```
>  mmchconfig pagepool=4G
> ```

maxFilesToCache
> The total number of different files that can be cached at one time. Every entry in the file cache requires some pageable memory to hold the content of the file's inode plus control data structures. This is in addition to any of the file's data and indirect blocks that might be cached in the page pool.
>
> The total amount of memory required for inodes and control data structures can be estimated as:
>
> ```
> maxFilesToCache × 3 KB
> ```
>
> Valid values of **maxFilesToCache** range from 1 to 100,000,000. For systems where applications use a large number of files, of any size, increasing the value for **maxFilesToCache** may prove beneficial. This is particularly true for systems where a large number of small files are accessed. The value should be large enough to handle the number of concurrently open files plus allow caching of recently used files.
>
> If the user does not specify a value for **maxFilesToCache**, the default value is 4000.

maxStatCache
> This parameter sets aside additional pageable memory to cache attributes of files that are not currently in the regular file cache. This is useful to improve the performance of both the system and GPFS **stat()** calls for applications with a working set that does not fit in the regular file cache.

For systems where applications test the existence of files, or the properties of files without actually opening them, as backup applications do, increasing the value for **maxStatCache** can improve performance.

The memory occupied by the stat cache can be calculated as:

```
maxStatCache × 400 bytes
```

The valid range for **maxStatCache** is 0 - 100,000,000. If you do not specify values for **maxFilesToCache** and **maxStatCache**, the default value of **maxFilesToCache** is 4000 and the default value of **maxStatCache** is 1000. If you specify a value for **maxFilesToCache** but not for **maxStatCache**, the default value of **maxStatCache** is 4 * **maxFilesToCache**.

Note: The stat cache is not effective on the Linux platform unless the Local Read-Only Cache (LROC) is configured. Therefore on Linux, the setting for **maxStatCache** depends on whether the Local Read-Only Cache (LROC) is enabled on the node:

- If LROC is not enabled on the node, set **maxStatCache** to 0.
- If LROC is enabled on the node, set **maxStatCache** to an appropriate nonzero value based on the number of nodes in the cluster, the number of token managers in the cluster, the size of the LROC device, and any other relevant factors. For more information about LROC, see the topic *Local read-only cache* in the *IBM Spectrum Scale: Administration Guide*.

This restriction applies to all the versions and distributions of Linux that IBM Spectrum Scale supports.

The total amount of memory GPFS uses to cache file data and metadata is arrived at by adding **pagepool** to the amount of memory required to hold inodes and control data structures (**maxFilesToCache** × 3 KB), and the memory for the stat cache (**maxStatCache** × 400 bytes) together. The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory on a node running GPFS.

During configuration, you can specify the **maxFilesToCache**, **maxStatCache**, and **pagepool** attributes that control how much cache is dedicated to GPFS. These values can be changed later, so experiment with larger values to find the optimum cache size that improves GPFS performance without negatively affecting other applications.

The **mmchconfig** command can be used to change the values of **maxFilesToCache**, **maxStatCache**, and **pagepool**. The **pagepool** parameter is the only one of these parameters that may be changed while the GPFS daemon is running. A change to the **pagepool** attribute occurs immediately when using the **-i** option on the **mmchconfig** command. Changes to the other values are effective only after the daemon is restarted.

For further information on these cache settings for GPFS, refer to *GPFS and memory* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The GPFS token system's effect on cache settings

Lock tokens play a role in maintaining cache consistency between nodes.

A token allows a node to cache data it has read from disk, because the data cannot be modified elsewhere without revoking the token first. Each token manager can handle approximately 300,000 different file tokens (this number depends on how many distinct byte-range tokens are used when multiple nodes access the same file). If you divide the 300,000 by the number of nodes in the GPFS cluster you get a value that should approximately equal **maxFilesToCache** (the total number of different files that can be cached at one time) + **maxStatCache** (additional pageable memory to cache file attributes that are not currently in the regular file cache).

Note the following about **maxFilesToCache** and **maxStatCache**:

- The valid range for **maxStatCache** is 0 - 100,000,000. If you do not specify values for **maxFilesToCache** and **maxStatCache**, the default value of **maxFilesToCache** is 4000 and the default value of **maxStatCache** is 1000. If you specify a value for **maxFilesToCache** but not for **maxStatCache**, the default value of **maxStatCache** is 4 * **maxFilesToCache**. On upgrades to IBM Spectrum Scale v4.1 from v3.4 or earlier, the existing defaults (1000 for **maxFilesToCache** and 4000 for **maxStatCache**) remain in effect.

  Note: The stat cache is not effective on the Linux platform unless the Local Read-Only Cache (LROC) is configured. Therefore on Linux, the setting for **maxStatCache** depends on whether the Local Read-Only Cache (LROC) is enabled on the node:
  - If LROC is not enabled on the node, set **maxStatCache** to 0.
  - If LROC is enabled on the node, set **maxStatCache** to an appropriate nonzero value based on the number of nodes in the cluster, the number of token managers in the cluster, the size of the LROC device, and any other relevant factors. For more information about LROC, see the topic *Local read-only cache* in the *IBM Spectrum Scale: Administration Guide*.

  This restriction applies to all the versions and distributions of Linux that IBM Spectrum Scale supports.
- **maxStatCache** can be set higher on user-interactive nodes and smaller on dedicated compute nodes, since **ls -l** performance is mostly a human response issue.
- **maxFilesToCache** should be large enough to handle the number of concurrently open files plus allow caching of recently used files.
- On upgrades to IBM Spectrum Scale v4.1 from v3.4 or earlier, the existing defaults (1000 for **maxFilesToCache** and 4000 for **maxStatCache**) remain in effect.
- **maxFilesToCache** and **maxStatCache** are indirectly affected by the **distributedTokenServer** configuration parameter because distributing the tokens across multiple token servers might allow keeping more tokens than if a file system has only one token server.

Access patterns

GPFS attempts to recognize the pattern of accesses (such as strided sequential access) that an application makes to an open file. If GPFS recognizes the access pattern, it will optimize its own behavior.

For example, GPFS can recognize sequential reads and will retrieve file blocks before they are required by the application. However, in some cases GPFS does not recognize the access pattern of the application or cannot optimize its data transfers. In these situations, you may improve GPFS performance if the application explicitly discloses aspects of its access pattern to GPFS through the **gpfs_fcntl()** library call.

Aggregate network interfaces

It is possible to aggregate multiple physical Ethernet interfaces into a single virtual interface. This is known as *Channel Bonding* on Linux and *EtherChannel/IEEE 802.3ad Link Aggregation* on AIX.

GPFS supports using such aggregate interfaces. The main benefit is increased bandwidth. The aggregated interface has the network bandwidth close to the total bandwidth of all its physical adapters. Another benefit is improved fault tolerance. If a physical adapter fails, the packets are automatically sent on the next available adapter without service disruption.

EtherChannel and IEEE802.3ad each requires support within the Ethernet switch. Refer to the product documentation for your switch to determine if EtherChannel is supported.

For details on how to configure EtherChannel and IEEE 802.3ad Link Aggregation and verify whether the adapter and the switch are operating with the correct protocols for IEEE 802.3ad, consult the operating system documentation.

Hint: Make certain that the switch ports are configured for **LACP** (the default is **PAGP**).

For additional service updates regarding the use of EtherChannel:

1. Go to the IBM Support Portal (www.ibm.com/support)
2. In the **Search** box, enter the search term *EtherChannel*
3. Click **Search**

Hint: A useful command for troubleshooting, where device is the Link Aggregation device, is:

`entstat -d device`

Swap space

It is important to configure a swap space that is large enough for the needs of the system.

While the actual configuration decisions should be made taking into account the memory requirements of other applications, it is a good practice to configure at least as much swap space as there is physical memory on a given node.

Linux configuration and tuning considerations

Configuration and tuning considerations for the Linux nodes in your system include the use of the **updatedb** utility, the **vm.min_free_kbytes** kernel tunable, and several other options that can improve GPFS performance.

For the latest system configuration and tuning settings, see the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) and the IBM Spectrum Scale Wiki (www.ibm.com/developerworks/community/wikis/home/wiki/General Parallel File System (GPFS)).

For more configuration and tuning considerations for Linux nodes, see the following topics:
1. "updatedb considerations"
2. "Memory considerations"
3. "GPFS helper threads" on page 42
4. "Communications I/O" on page 42
5. "Disk I/O" on page 42

updatedb considerations

On some Linux distributions, the system is configured by default to run the file system indexing utility **updatedb** through the **cron** daemon on a periodic basis (usually daily).

This utility traverses the file hierarchy and generates a large I/O load. For this reason, it is configured by default to skip certain file system types and nonessential file systems. However, the default configuration does not prevent **updatedb** from traversing GPFS file systems. In a cluster this results in multiple instances of **updatedb** traversing the same GPFS file system simultaneously. This causes general file system activity and lock contention in proportion to the number of nodes in the cluster. On smaller clusters, this may result in a relatively short-lived spike of activity, while on larger clusters, depending on the overall system throughput capability, the period of heavy load may last longer. Usually the file system manager node will be the busiest, and GPFS would appear sluggish on all nodes. Re-configuring the system to either make **updatedb** skip all GPFS file systems or only index GPFS files on one node in the cluster is necessary to avoid this problem.

Memory considerations

It is recommended that you adjust the **vm.min_free_kbytes** kernel tunable. This tunable controls the amount of free memory that Linux kernel keeps available (that is, not used in any kernel caches).

When **vm.min_free_kbytes** is set to its default value, on some configurations it is possible to encounter memory exhaustion symptoms when free memory should in fact be available. Setting **vm.min_free_kbytes** to 5-6% of the total amount of physical memory, but no more than 2 GB, can prevent this problem.

See the GPFS Redbooks® papers for more information:
- *GPFS Sequential Input/Output Performance on IBM pSeries 690* (www.redbooks.ibm.com/redpapers/pdfs/redp3945.pdf)
- *Native GPFS Benchmarks in an Integrated p690/AIX and x335/Linux Environment* (www.redbooks.ibm.com/redpapers/pdfs/redp3962.pdf)

GPFS helper threads

GPFS uses helper threads such as **prefetchThreads** and **workerThreads** to improve performance.

Since systems vary, it is suggested you simulate an expected workload in GPFS and examine available performance indicators on your system. For instance some SCSI drivers publish statistics in the **/proc/scsi** directory. If your disk driver statistics indicate that there are many *queued requests* it may mean you should throttle back the helper threads in GPFS.

For more information, see *Parameters for performance tuning and optimization* in *IBM Spectrum Scale: Administration Guide*.

Communications I/O

Values suggested here reflect evaluations made at the time this documentation was written. For the latest system configuration and tuning settings, see the IBM Spectrum Scale Wiki (www.ibm.com/developerworks/community/wikis/home/wiki/General Parallel File System (GPFS)).

To optimize the performance of GPFS and your network, it is suggested you do the following:
- Enable Jumbo Frames if your switch supports it.

  If GPFS is configured to operate over Gigabit Ethernet, set the MTU size for the communication adapter to 9000.
- Verify **/proc/sys/net/ipv4/tcp_window_scaling** is enabled. It should be by default.
- Tune the TCP window settings by adding these lines to the **/etc/sysctl.conf** file:
```
# increase Linux TCP buffer limits
net.core.rmem_max = 8388608
net.core.wmem_max = 8388608
# increase default and maximum Linux TCP buffer sizes
net.ipv4.tcp_rmem = 4096 262144 8388608
net.ipv4.tcp_wmem = 4096 262144 8388608
```
  After these changes are made to the **/etc/sysctl.conf** file, apply the changes to your system:
  1. Issue the **sysctl -p /etc/sysctl.conf** command to set the kernel settings.
  2. Issue the **mmshutdown -a** command and then issue **mmstartup -a** command to restart GPFS

Disk I/O

To optimize disk I/O performance, you should consider the following options for NSD servers or other GPFS nodes that are directly attached to a SAN over a Fibre Channel (FC) network.
1. The storage server cache settings can impact GPFS performance if not set correctly.
2. When the storage server disks are configured for RAID5, some configuration settings can affect GPFS performance. These settings include:
   - GPFS block size
   - Maximum I/O size of the Fibre Channel host bus adapter (HBA) device driver
   - Storage server RAID5 stripe size

Note: For optimal performance, GPFS block size should be a multiple of the maximum I/O size of the FC HBA device driver. In addition, the maximum I/O size of the FC HBA device driver should be a multiple of the RAID5 stripe size.

3. These suggestions may avoid the performance penalty of read-modify-write at the storage server for GPFS writes. Examples of the suggested settings are:
   * 8+P RAID5
     – GPFS block size = 512K
     – Storage Server RAID5 segment size = 64K (RAID5 stripe size=512K)
     – Maximum IO size of FC HBA device driver = 512K
   * 4+P RAID5
     – GPFS block size = 256K
     – Storage Server RAID5 segment size = 64K (RAID5 stripe size = 256K)
     – Maximum IO size of FC HBA device driver = 256K

   For the example settings using 8+P and 4+P RAID5, the RAID5 parity can be calculated from the data written and will avoid reading from disk to calculate the RAID5 parity. The maximum IO size of the FC HBA device driver can be verified using **iostat** or the Storage Server performance monitor. In some cases, the device driver may need to be patched to increase the default maximum IO size.

4. The GPFS parameter **maxMBpS** can limit the maximum throughput of an NSD server or a single GPFS node that is directly attached to the SAN with a FC HBA. The default value is 2048. The **maxMBpS** parameter is changed by issuing the **mmchconfig** command. If this value is changed, restart GPFS on the nodes, and test the read and write performance of a single node and a large number of nodes.

AIX configuration and tuning considerations

For the latest system configuration settings, see the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

GPFS use with Oracle

When using GPFS with Oracle, configuration and tuning considerations include the following.

* When setting up your LUNs, it is important to create the NSD such that they map one-to-one with a LUN that is a single RAID device.
* For file systems holding large Oracle databases, set the GPFS file system block size through the **mmcrfs** command using the **-B** option, to a large value:
  – 512 KB is generally suggested.
  – 256 KB is suggested if there is activity other than Oracle using the file system and many small files exist which are not in the database.
  – 1 MB is suggested for file systems 100 TB or larger.

  The large block size makes the allocation of space for the databases manageable and has no affect on performance when Oracle is using the Asynchronous I/O (AIO) and Direct I/O (DIO) features of AIX.
* Set the GPFS worker threads through the **mmchconfig worker1Threads** command to allow the maximum parallelism of the Oracle AIO threads.
  – Adjust the GPFS prefetch threads accordingly through the **mmchconfig prefetchThreads** command. The maximum value of **prefetchThreads** plus **worker1Threads** plus **nsdMaxWorkerThreads** is 8192 on all 64-bit platforms.
  – When requiring GPFS sequential I/O, set the prefetch threads between 50 and 100 (the default is 72).

  Note: These changes through the **mmchconfig** command take effect upon restart of the GPFS daemon.

- The number of AIX AIO *kprocs* to create should be approximately the same as the GPFS **worker1Threads** setting.
- The AIX AIO *maxservers* setting is the number of *kprocs* PER CPU. It is suggested to set is slightly larger than the value of **worker1Threads** divided by the number of CPUs. For example if **worker1Threads** is set to 500 on a 32-way SMP, set *maxservers* to 20.
- Set the Oracle database block size equal to the LUN segment size or a multiple of the LUN pdisk segment size.
- Set the Oracle read-ahead value to prefetch one or two full GPFS blocks. For example, if your GPFS block size is 512 KB, set the Oracle blocks to either 32 or 64 16 KB blocks.
- Do not use the **dio** option on the **mount** command as this forces DIO when accessing *all* files. Oracle automatically uses DIO to open database files on GPFS.
- When running Oracle RAC 10g, it is suggested you increase the value for **OPROCD_DEFAULT_MARGIN** to at least 500 to avoid possible random reboots of nodes.

  In the control script for the Oracle CSS daemon, located in **/etc/init.cssd** the value for **OPROCD_DEFAULT_MARGIN** is set to 500 (milliseconds) on all UNIX derivatives except for AIX. For AIX this value is set to 100. From a GPFS perspective, even 500 milliseconds maybe too low in situations where node failover may take up to a minute or two to resolve. However, if during node failure the surviving node is already doing direct IO to the **oprocd** control file, it should have the necessary tokens and indirect block cached and should therefore not have to wait during failover.

Chapter 4. Parameters for performance tuning and optimization

Use these parameters with the `mmchconfig` command for performance tuning and optimization.

Tuning guide for frequently-changed parameters

autoload
  When **autoload** is set to yes, GPFS starts automatically on the nodes that are rebooted. The rebooted nodes rejoin the cluster, the `file system automount` option is set to **yes**, and the file system is mounted. The default value of this parameter is **no**.

  Important: Set **autoload** to **no** before fixing hardware issues and performing system maintenance.

deadlockDetectionThreshold
  When **deadlockDetectionThreshold** is set to 0, the GPFS dead-lock detection feature is disabled. The default value of this parameters is 300 seconds.

  Important: You must enable the GPFS dead-lock detection feature to collect debug data and resolve dead lock issue in a cluster. If dead-lock events occur frequently, fix the problem instead of disabling the feature.

defaultHelperNodes
  The nodes that are added to **defaultHelperNodes** are used in running certain commands, such as **mmrestripefs**. Running the GPFS command on partial nodes in a cluster, such as running the mmrestripefs command on all NSD server nodes, might have a better performance. The default value of this parameter is all nodes in cluster.

  Important: Set the –N option for GPFS management commands or change the value of **defaultHelperNodes** before running the GPFS management commands.

maxFilesToCache
  The **maxFilesToCache** parameter specifies the number of files that can be cached by each node. The range of valid values for **maxFilesToCache** is 1 - 100,000,000. The default value is 4000. The value of this parameter must be large enough to handle the number of concurrently open files and to allow the caching of recently used files.

  Changing the value of **maxFilesToCache** affects the amount of memory that is used on the node. In a large cluster, a change in the value of **maxFilesToCache** is greatly magnified. Increasing **maxFilesToCache** in a large cluster with hundreds of nodes increases the number of tokens a token manager needs to store. Ensure that the manager node has enough memory and **tokenMemLimit** is increased when you are running GPFS version 4.1.1 and earlier. Therefore, increasing the value of **maxFilesToCache** on large clusters usually happens on a subset of nodes that are used as log-in nodes, SMB and NFS exporters, email servers, and other file servers.

  For systems on which applications use a large number of files, increasing the value of **maxFilesToCache** might be beneficial, especially where a large number of small files are accessed.

  Note: Setting the **maxFilesToCache** parameter to a high value results in a large amount of memory being allocated for internal data buffering. If the value of **maxFilesToCache** is set too high, some operations in IBM Spectrum Scale might not have enough memory to run in. If you have set **maxFilesToCache** to a very high value and you see error messages in the `mmfs.log` file that say that not enough memory is available to perform an operation, try lowering the value of **maxFilesToCache**.

maxBlockSize
  The value of **maxBlockSize** must be equal to or larger than the maximum block size of all the file

systems in the local and remote clusters. Before changing this parameter, ensure that the GPFS daemon on each node in the cluster is shut down. The default value is 4 MB.

Note: When you migrate a cluster from an earlier version to version 5.0.0 or later, the value of **maxblocksize** stays the same. However, if **maxblocksize** was set to **DEFAULT** in the earlier version of the cluster, then migrating it to version 5.0.0 or later sets it explicitly to 1 MiB, which was the default size in earlier versions. To change maxBlockSize to the default size after migrating to version 5.0.0 or later, set **maxblocksize**=DEFAULT (4 MiB).

For more information, see the topics *mmcrfs* and *mmchconfig* in the *IBM Spectrum Scale: Command and Programming Reference*.

maxMBpS

The **maxMBpS** parameter indicates the maximum throughput in megabytes per second that GPFS can submit into or out of a single node. GPFS calculates from this variable how many prefetch/writebehind threads to schedule for sequential file access.

In GPFS version 3.5 and earlier, the default value is 2048. But if the node has faster interconnect, such as InfiniBand or 40GigE or multiple links) you can set the parameter to a higher value. As a general rule, try setting **maxMBpS** to twice the I/O throughput that the node can support. For example, if the node has 1 x FDR link and the GPFS configuration parameter **verbRdma** has been enabled, then the expected throughput of the node is 6000 MB/s. In this case, set **maxMBpS** to 12000.

Setting **maxMBpS** does not guarantee the desired GPFS sequential bandwidth on the node. All the layers of the GPFS stack, including the node, the network, and the storage subsystem, must be designed and tuned to meet the I/O performance requirements.

maxStatCache

The **maxStatCache** parameter sets aside the pageable memory to cache attributes of files that are not currently in the regular file cache. This improves the performance of stat() calls for applications with a working set that does not fit in the regular file cache. For systems where applications test the existence of files, or the properties of files, without actually opening them as backup applications do, increasing the value for **maxStatCache** can be beneficial.

The valid range for **maxStatCache** is 0 - 100,000,000. If you do not specify values for **maxFilesToCache** and **maxStatCache**, the default value of **maxFilesToCache** is 4000 and the default value of **maxStatCache** is 1000. If you specify a value for **maxFilesToCache** but not for **maxStatCache**, the default value of **maxStatCache** is 4 * **maxFilesToCache**.

Note: The stat cache is not effective on the Linux platform unless the Local Read-Only Cache (LROC) is configured. Therefore on Linux, the setting for **maxStatCache** depends on whether the Local Read-Only Cache (LROC) is enabled on the node:

- If LROC is not enabled on the node, set **maxStatCache** to 0.
- If LROC is enabled on the node, set **maxStatCache** to an appropriate nonzero value based on the number of nodes in the cluster, the number of token managers in the cluster, the size of the LROC device, and any other relevant factors. For more information about LROC, see the topic *Local read-only cache* in the *IBM Spectrum Scale: Administration Guide*.

This restriction applies to all the versions and distributions of Linux that IBM Spectrum Scale supports.

nsdMaxWorkerThreads

NSD server tuning. For more information about GPFS NSD server design and tuning, see NSD Server Design and Tuning.

pagepool

The **pagepool** parameter is used to change the size of the data cache on each node. The default value is either one-third of the physical memory of the node or 1G, whichever is smaller. This value applies to new installations only. On upgrades, the existing default value is maintained.

The maximum GPFS **pagepool** size depends on the value of the **pagepoolMaxPhysMemPct** parameter and the amount of physical memory on the node. Unlike local file systems that use the operating system page cache to cache file data, GPFS allocates its own cache called the page pool. The GPFS page pool is used to cache user file data and file system metadata. Along with file data, the page pool supplies memory for various types of buffers such as prefetch and write behind. The default page pool size might be sufficient for sequential IO workloads. The default page pool size might not be sufficient for Random IO or workloads that involve a large number of small files.

In some cases, allocating 4 GB, 8 GB, or more memory can improve the workload performance. For database applications that use Direct IO, the page pool is not used for any user data. The main purpose in this case is for system metadata and caching the indirect blocks for the files. For NSD server, if no applications or file system manager services are running on NSD server, the page pool is only used transiently by the NSD worker threads to gather data from client nodes and write the data to disk. The NSD server does not cache any of the data.

readReplicaPolicy

The **readReplicaPolicy** parameter specifies the location from which the disk must read the replicas. The valid values are default, local and fastest. The default value is default.

By default, GPFS reads the first replica even when there is no replica on the local disk. When the value of this parameter is set to local, the policy reads replicas from the local disk only if the local disk has data. For performance considerations, this is the recommended setting for FPO environments. When the value of this parameter is set to fastest, the policy reads replicas from the disk considering the fastest based on the read I/O statistics of the disk. In a system with SSD and regular disks, the value of **fastestPolicyCmpThreshold** can be set to a greater number, such as 100, to let GPFS refresh the slow disk speed statistics less frequently.

restripeOnDiskFailure

The **restripeOnDiskFailure** specifies if GPFS attempts to automatically recover from certain common disk failure situations. The default value of this parameter is no.

Important: While deploying FPO or when the HAWC feature is enabled, set the **restripeOnDiskFailure** parameter to yes.

tiebreakerDisks

For a small cluster with up to eight nodes that have SAN-attached disk systems, define all nodes as quorum nodes and use tiebreaker disks. With more than eight nodes, use only node quorum. While defining the tiebreaker disks, you can use the SAN-attached NSD in the file system. The default value of this parameter is null, which means no tiebreaker disk has been defined.

unmountOnDiskFail

The **unmountOnDiskFail** specifies how the GPFS daemon responds when a disk failure is detected. The valid values of this parameter are yes, no, and meta. The default value is no.

Important: Set the value of **unmountOnDiskFail** to meta for FPO deployment or when the file system metadata and data replica are more than one.

workerThreads

The **workerThreads** parameter controls an integrated group of variables that tune the file system performance in environments that are capable of high sequential and random read and write workloads and small file activity.

The default value of this parameter is 48 for a base IBM Spectrum Scale cluster and 512 for a cluster with protocols installed. A valid value can be any number ranging from 1 to 8192. The -N flag is valid with this variable. This variable controls both internal and external variables. The internal variables include maximum settings for concurrent file operations, for concurrent threads that flush dirty data and metadata, and for concurrent threads that prefetch data and metadata. You can adjust the following external variables with the mmchconfig command:

- *logBufferCount*

- *preFetchThreads*
- *worker3Threads*

Tuning parameters change history

Parameter	3.5	4.1	4.1.0.4	4.1.1	4.2	4.2.1	4.2.2	4.2.3	5.0.0
adminMode									
afmAsyncDelay	Updated	Updated							
afmAsyncOpWaitTimeout									Added
afmDirLookupRefreshInterval									
afmDirOpenRefreshInterval									
afmDisconnectTimeout									
afmExpirationTimeout									
afmFileLookupRefreshInterval									
afmFileOpenRefreshInterval									
afmHardMemThreshold	Not applicable	Not applicable	Not applicable	Not applicable	Added				
afmHashVersion	Not applicable	Added							Updated
afmMaxParallelRecoveries									Added
afmNumReadThreads	Not applicable	Added	Updated						
afmNumWriteThreads	Not applicable	Not applicable	Added						
afmParallelReadChunkSize	Not applicable	Added							
afmParallelReadThreshold	Not applicable	Added							
afmParallelWriteChunkSize	Not applicable	Added							
afmParallelWriteThreshold	Not applicable	Added							
afmReadSparseThreshold	Updated								
afmRevalOpWaitTimeout									Added
afmSecondaryRW	Not applicable	Not applicable	Not applicable	Not applicable	Added				
afmShowHomeSnapshot									
afmSyncOpWaitTimeout									Added
atimeDeferredSeconds									
autoload		Updated							
automountDir		Updated							
cesSharedRoot	Not applicable	Not applicable	Not applicable	Added		Updated			
cifsBypassTraversalChecking	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
cipherList			Updated						
cnfsGrace	Not applicable	Added							
cnfsMountdPort									
cnfsNFSDprocs									

Parameter	3.5	4.1	4.1.0.4	4.1.1	4.2	4.2.1	4.2.2	4.2.3	5.0.0
cnfsReboot	Not applicable	Added							
cnfsSharedRoot		Updated							
cnfsVersions	Not applicable	Added							
cnfsVIP			Obsoleted						
commandAudit	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
dataDiskCacheProtectionMethod	Added					Updated			
dataDiskWaitTimeForRecovery									
dataStructureDump						Updated			
deadlockBreakupDelay	Not applicable	Added							
deadlockDataCollectionDailyLimit	Not applicable	Added			Updated	Updated		Updated	
deadlockDataCollectionMinInterval	Not applicable	Not applicable	Not applicable	Added	Updated	Updated			
deadlockDetectionThreshold	Not applicable	Added			Updated	Updated			
deadlockDetectionThreshold-ForShortWaiters	Not applicable	Not applicable	Not applicable	Added					
deadlockDetectionThreshold-IfOverloaded	Not applicable	Not applicable	Not applicable	Added	Updated	Obso-leted			
deadlockOverloadThreshold	Not applicable	Not applicable	Not applicable	Added	Updated	Updated			
debugDataControl	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added		Updated	
defaultHelperNodes			Updated						
defaultMountDir									
disableInodeUpdateOnFdatasync	Added								
dmapiDataEventRetry									
dmapiEventTimeout								Updated	
dmapiMountEvent									
dmapiMountTimeout									
dmapiSessionFailureTimeout									
enbleIPv6									
enforceFilesetQuotaOnRoot									
expelDataCollectionDailyLimit	Not applicable	Not applicable	Not applicable	Added					
expelDataCollectionMinInterval	Not applicable	Not applicable	Not applicable	Added					
failureDetectionTime									
fastestPolicyCmpThreshold	Not applicable	Not applicable	Not applicable	Added		Updated			
fastestPolicyMaxValidPeriod	Not applicable	Not applicable	Not applicable	Added					
fastestPolicyMinDiffPercent	Not applicable	Not applicable	Not applicable	Added					
fastestPolicyNumReadSamples	Not applicable	Not applicable	Not applicable	Added					
fileHeatLossPercent									

Parameter	3.5	4.1	4.1.0.4	4.1.1	4.2	4.2.1	4.2.2	4.2.3	5.0.0
fileHeatPeriodMinutes									
FIPS1402mode	Not applicable	Added	Updated						
forceLogWriteOnFdatasync	Added								
ignorePrefetchLUNCount	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added		Updated	
lrocData	Not applicable	Added							
lrocDataMaxFileSize	Not applicable	Added							
lrocDataStubFileSize	Not applicable	Added							
lrocDirectories	Not applicable	Added							
lrocEnableStoringClearText									Added
lrocInodes	Not applicable	Added							
maxblocksize	Updated		Updated						Updated
maxBufferDescs	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
maxDownDisksForRecovery	Not applicable	Not applicable	Not applicable	Added					
maxFailedNodesForRecovery	Not applicable	Not applicable	Not applicable	Added					
maxFcntlRangesPerFile									
maxFilesToCache									
maxMBpS									
maxMissedPingTimeout	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
maxStatCache				Updated				Updated	
metadataDiskWaitTimeForRecovery			Updated						
minDiskWaitTimeForRecovery	Not applicable	Not applicable	Not applicable	Added					
minMissedPingTimeout	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
mmapRangeLock									
mmfsLogTimeStampISO8601							Added		
nfsPrefetchStrategy	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
nistCompliance	Not applicable	Added							
noSpaceEventInterval									
nsdBufSpace									
nsdInlineWriteMax	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
nsdMaxWorkerThreads	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
nsdMinWorkerThreads	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
nsdMultiQueue	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
nsdRAIDTracks									

Parameter	3.5	4.1	4.1.0.4	4.1.1	4.2	4.2.1	4.2.2	4.2.3	5.0.0
nsdRAIDBufferPoolSizePct									
nsdServerWaitTimeForMount									
nsdServerWaitTimeWindowOnMount									
numaMemoryInterleave	Not applicable	Not applicable	Added						
pagepool					Updated				
pagepoolMaxPhysMemPct									
pitWorkerThreadsPerNode	Not applicable	Not applicable	Not applicable	Added					
prefetchPct	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
prefetchThreads			Updated						
profile	Not applicable	Not applicable	Not applicable	Added					
readReplicaPolicy	Not applicable	Not applicable	Not applicable	Added		Updated			
release=LATEST						Updated			
restripeOnDiskFailure						Updated			
rpcPerfNumberDayIntervals	Not applicable	Added							
rpcPerfNumberHourIntervals	Not applicable	Added							
rpcPerfNumberMinuteIntervals	Not applicable	Added							
rpcPerfNumberSecondIntervals	Not applicable	Added							
rpcPerfRawExecBufferSize	Not applicable	Added							Updated
rpcPerfRawStatBufferSize	Not applicable	Added							Updated
seqDiscardThreshold	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
sidAutoMapRangeLength									
sidAutoMapRange									
subnets									
sudoUser									Added
syncBuffsPerIteration	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
syncSambaMetadataOps	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			
systemLogLevel	Not applicable	Added							
tiebreakerDisks		Updated						Update	
uidDomain									
unmountOnDiskFail					Updated				
usePersistentReserve									
verbsPorts	Updated						Updated		Updated
verbsRdma									
verbsRdmaCm	Added								
verbsRdmaPkey	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added	

Parameter	3.5	4.1	4.1.0.4	4.1.1	4.2	4.2.1	4.2.2	4.2.3	5.0.0
verbsRdmaRoCEToS	Not applicable	Not applicable	Added						
verbsRdmaSend									
verbsRdmasPerConnection					Updated	Updated			Obsoleted
verbsRdmasPerNode						Updated			Obsoleted
verbsRecvBufferCount									Added
verbsRecvBufferSize									Added
verbsSendBufferMemoryMB									Obsoleted
workerThreads	Not applicable	Not applicable	Not applicable	Not applicable	Added		Updated		
worker1Threads					Updated				
writebehindThreshold	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Added			

For more information on the configuration attributes, see "Changing the GPFS cluster configuration data" on page 4.

Chapter 5. Ensuring high availability of the GUI service

You need multiple GUI nodes configured in the system to ensure high availability of the GUI service. You also need to set up a cluster configure repository (CCR) when you plan to configure multiple GUI nodes in the cluster. The CCR is used to store certain important configuration details that must be shared among all GUI nodes.

The following figure illustrates the GUI high availability configuration with two GUI nodes.
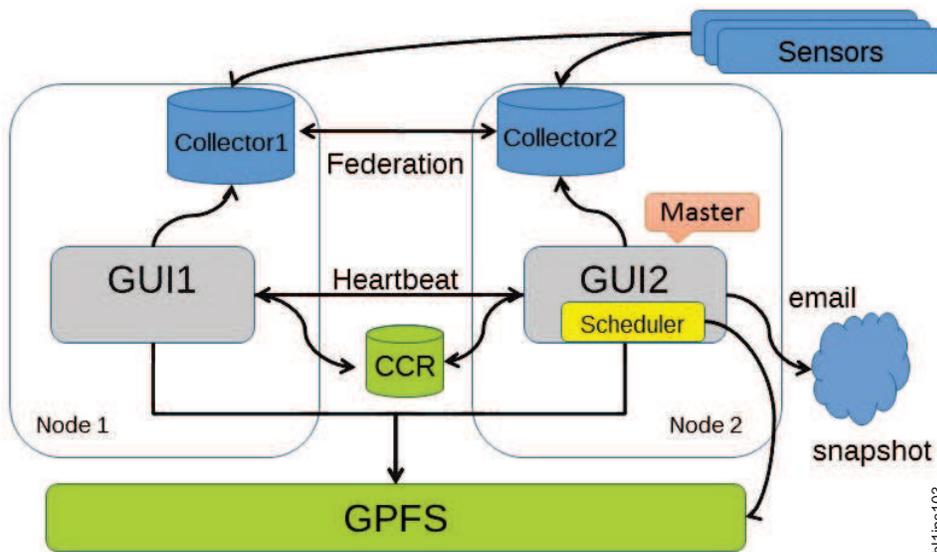
Figure 1. High availability configuration of GUI with two GUI nodes

The following list provides the configuration requirements to ensure high availability of the GUI service:
- Up to three GUI nodes can be configured in a cluster. Perform the following steps to set up a GUI node:
  - Install the GUI package on the node. For more information on the latest packages that are required for different platforms, see *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
  - Start the GUI service and either log in or run `/usr/lpp/mmfs/gui/cli/initgui` to initialize the GUI database. Now, the GUI becomes fully functional and it adds the node to the GUI_MGMT_SERVERS node class.
- The GUI nodes are configured in the active/active configuration. All GUI nodes are fully functional and can be used in parallel.
- Each GUI has its own local configuration cache in PostgreSQL and collects configuration changes individually.
- One GUI node is elected as the master node. This GUI instance exclusively performs some tasks that must be run only once in a cluster such as, running snapshot schedules and sending email and SNMP notifications. If services that are run on the master GUI node are configured, the environment for all the GUI nodes must support these services on all nodes. For example, it needs to be ensured that access to SMTP and SNMP servers is possible from all GUI nodes and not only from the master GUI node. You can use the following utility function, which displays the current master GUI node:

```
[root@gpfsgui-11 ~]# /usr/lpp/mmfs/gui/cli/lsnode
Hostname          IP          Description    Role            Product  Connection GPFS     Last updated
                                                             version  status     status
```

```
gpfsgui-11.novalocal 10.0.100.12 Master GUI Node management,storage 5.0.0.0 HEALTHY    HEALTHY 7/10/17 10:19 AM
gpfsgui-12.novalocal 10.0.100.13                  storage,ces      5.0.0.0 HEALTHY    HEALTHY 7/10/17 10:19 AM
gpfsgui-13.novalocal 10.0.100.14                  storage,ces      5.0.0.0 HEALTHY    HEALTHY 7/10/17 10:19 AM
```

- All GUI nodes are equal from the user's perspective. If a GUI node fails, the user must manually connect to the other GUI. The master role fails over automatically. But there is no failover for the IP address of the other GUI server.
- Data that cannot be gathered from GPFS is stored in CCR as shared-cluster repository. This includes GUI users, groups and roles, snapshot schedules, email notification settings, policy templates, and ACL templates.
- All GUI nodes must run on the same software level.
- If an external authentication method such as AD or LDAP is used to store the GUI user details and authenticate them, you must configure AD/LDAP on all GUI nodes to ensure high-availability. If internal authentication method is used, the GUI nodes get the user information from the CCR.
- To display the performance monitoring information, install performance monitoring collector on each GUI node and these collectors must be configured in the federated mode. The data collection from the sensors can be configured in such a way that the details are sent either to all collectors or only to a single collector.
- The **Mark as Read** operation can be performed on events that are stored locally on the GUI node. The changes that are made to the events are not visible through the other GUI node.
- Each GUI has its own local configuration cache and collects configuration changes individually.
- A corrupted cache database affects only the local GUI. Other GUIs continue working. Most of the configuration changes are simultaneously reported in the GUI. Some configuration changes are gathered through the individually scheduled refresh tasks, which might result in displaying unsynchronized information.

Chapter 6. Configuring and tuning your system for Cloud services

This topic describes the procedure for configuring and tuning your IBM Spectrum Scale node for Cloud services.

Designating the Cloud services nodes

This topic describes how to designate a node as Cloud services node in the IBM Spectrum Scale cluster.

Before you begin, ensure that you install the server package RPMs on all nodes that you want to designate as Cloud services nodes. These nodes must have GPFS server licenses enabled.

Also, ensure that a user-defined node class is created and properly configured for Cloud services. For instructions, see *Creating a user-defined node class for Transparent cloud tieringorCloud data sharing* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

To start working with Cloud services, the administrator first needs to designate a node as Cloud services node in the IBM Spectrum Scale cluster. Data migration to or data recall from a cloud object storage occurs in this node.

You can designate a maximum of any combination of 4 CES or NSD nodes as Cloud services nodes in each node class (with a maximum of four node class for 16 nodes total) in the IBM Spectrum Scale cluster.

If you use multiple node classes for Cloud services, then you can designate at least one node in each node class as Cloud services server nodes.

By default and by way of recommendation, Cloud services use the node IP addresses, not the CES IPs.

Note: You need to perform this procedure only on a single node where the server package is installed.

1. To designate the nodes as Cloud services nodes, issue a command according to this syntax: **`mmchnode change-options -N {Node[,Node...] | NodeFile | NodeClass} [--cloud-gateway-nodeclass CloudGatewayNodeClass]`**.

   You can either choose to designate all nodes or only some selected nodes in a node class as Cloud services nodes.

   To designate all nodes in the node class, *TCTNodeClass1*, as Cloud services server nodes, issue this command:

   `mmchnode --cloud-gateway-enable -N TCTNodeClass1`

   To designate only a few nodes (node1 and node2) in the node class, *TCTNodeClass1*, as Cloud services server nodes, issue this command:

   `mmchnode --cloud-gateway-enable -N node1,node2 --cloud-gateway-nodeclass TCTNodeClass1`

   It designates only `node1` and `node2` as Cloud services server nodes from the node class, *TCTNodeClass1*. Administrators can continue to use the node class for other purposes.

   Note: The Cloud services node must have connectivity to the object storage service that the Cloud services uses.

2. To designate nodes from multiple node classes as Cloud services server nodes, issue the following commands:
   - `mmchnode --cloud-gateway-enable -N TCTNodeClass1`

- `mmchnode --cloud-gateway-enable -N TCTNodeClass2`

   | **Note:** These nodes cannot be combined into a single Cloud services across node classes because
   | Cloud services for nodes in different node classes are always different or separate.
   3. To list the designated Transparent cloud tiering nodes, issue this command: `mmcloudgateway node list`

   Note: For more information, see the `mmcloudgateway` command.
   4. To disable two nodes, *node1* and *node2*, from the node class, *TCTNodeClass1*, issue this command:
      `mmchnode --cloud-gateway-disable -N nod1,node2 --cloud-gateway-nodeclass TCTNodeClass1`

You can add a node to the node class at any time. For example, issue the following commands to add the node, *10.11.12.13*, to the node class, *TCTNodeClass1*.
   1. `mmchnodeclass` TCTNodeClass1 add -N 10.11.12.13
   2. `mmchnode` --cloud-gateway-enable -N 10.11.12.13 --cloud-gateway-nodeclass TCTNodeClass1

Starting up the Cloud services software

This topic describes how to start the Cloud services software on IBM Spectrum Scale nodes.

Before you try to start Cloud services on a node, ensure that the node is designated as a Cloud services node. For more information, see "Designating the Cloud services nodes" on page 55.

Start the Cloud services before you run any of the Cloud services commands.

To start Cloud services, issue a command according to this syntax:
| `mmcloudgateway service start [-N {alltct | Node[,Node...] | NodeFile | NodeClass}]`

For example, to start the service on all Transparent cloud tiering nodes in a cluster, issue this command:
`mmcloudgateway service start -N alltct`

To start the service on all Cloud services nodes as provided in the node class, *TCTNodeClass1*, issue this command:
`mmcloudgateway service start -N TCTNodeClass1`

If you provide this command without any arguments, the service is started on the current node.

If you have more than one node class, then you must start the Cloud services individually on each node class, as follows:
- `mmcloudgateway service start -N TCTNodeClass1`
- `mmcloudgateway service start -N TCTNodeClass2`

It is a good practice to verify that the service is started. Enter a command like the following one:
`mmcloudgateway service status -N TCTNodeClass`

Note: You can run this command from any node in the cluster, not necessarily from a node that is part of a node class.

| **Next step**: See "Managing a cloud storage account" on page 57.

Managing a cloud storage account

You can manage a cloud storage account by using the **mmcloudgateway account create/update/delete** command.

Note:
- Before you try to configure a cloud storage account, ensure that the Cloud services are started. For more information, see "Starting up the Cloud services software" on page 56.
- Before deleting a cloud storage account, ensure that you recall all the data that is migrated to the cloud.
- Ensure that Network Time Protocol (NTP) is enabled and time is correctly set.
- The following regions are supported for Amazon:
  - us-standard
  - us-west-1
  - us-west-2
  - eu-west-1
  - eu-central-1
  - sa-east-1
  - ap-southeast-1
  - ap-southeast-2
  - ap-south-1
  - ap-northeast-1
  - ap-northeast-2
- To configure a new cloud object storage account for the IBM Cloud Object Storage version 3.7.2 and above, enter a command like the following:

  ```
  mmcloudgateway account create --cloud-nodeclass TCTNodeClass1 --account-name
  cscloud --cloud-type CLEVERSAFE-NEW --username  user1  --pwd-file MyFile.txt
  ```

  The system displays output similar to this:

  ```
  mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
  mmcloudgateway: This may take a while...
  mmcloudgateway: Command completed successfully on vm1.gpfs.net.
  mmcloudgateway: You can now delete the password file 'MyFile.txt'
  mmcloudgateway: Command completed.
  ```

- To create a cloud account for deploying a WORM solution by using locked vaults, issue a command like the following:

  ```
  mmcloudgateway account create --cloud-nodeclass NodeClass1 --account-name myCloud
    --cloud-type CLEVERSAFE-NEW
    --src-keystore-path /root/test/testalias.ssl/testalias.jks
    --src-alias-name testalias --src-keystore-type JKS
    --src-keystore-pwd-file /root/pwd/file.txt
  ```

- To update a cloud account, issue a command like the following:

  ```
  mmcloudgateway account update  --cloud-nodeclass cloud --account-name new --username root
  ```

  The system displays output similar to this:

  ```
  mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
  mmcloudgateway: This may take a while...
  mmcloudgateway: Command completed successfully on vm1.gpfs.net.
  mmcloudgateway: Command completed.
  ```

- To delete a cloud account, issue a command like the following:

  ```
  mmcloudgateway account delete --cloud-nodeclass cloud --account-name new
  ```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with vmi.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmi.gpfs.net.
mmcloudgateway: Command completed.
```

Note: You can delete a cloud storage account when you move from one cloud account to another. For example, you have used Amazon S3 for testing purposes and then want to move to IBM Cloud Object Storage for production. In this case, you can delete the S3 account and then create a new IBM Cloud Object Storage.

Note: Even though you specify the credentials for the cloud account, the actual validation does not happen here. The authentication of the credentials happens only when you create a cloud storage access point. Therefore, you do not receive any authentication error even if you provide some wrong cloud account credentials.

Next step: See "Defining cloud storage access points (CSAP)."

Defining cloud storage access points (CSAP)

The Cloud Storage Access Point (CSAP) provides access between the cloud account on your object storage and IBM Spectrum Scale. You must create at least one CSAP per cloud account so your Cloud services has a path to the object storage. Extra CSAPs can also be created. CSAPs that all have about the same lowest latency (which is tested every 30 minutes) to a node are used evenly. CSAPs with higher latency are put in standby and are used only in error scenarios. This provides greater throughput and higher availability in various error scenarios.

You can send data to the cloud object storage via Cloud Storage Access Points (CSAPs). Each cloud account needs at least one CSAP defined in order to have a path to the cloud. For some cases (IBM SoftLayer®, Amazon S3, or cloud storage with a load balancer with built-in redundancy) one accessor suffices. However, for cases where traffic is going directly to the object storage, it is usually beneficial to have more than one CSAP to provide needed availability and bandwidth for performance. For example, if you are designing an on-premise solution with IBM Cloud Object Storage, you will want to create one access point for each accessor node you want to send data to. The Cloud services will randomly assign work to the available accessors as long as they are performing properly (broken or slow access points are avoided).

Note: If multiple intermediate certificates are issued by an internal certifying authority (CA), ensure to provide only a self-signed internal CA rather than providing a file that contains all the intermediate certificates. For example, if the CA issued a certificate chain such as `Internal CA->cert1->cert2`, then the input `pem` file should contain only the Internal CA certificate.

To create, update, or delete a CSAP:

- To create a CSAP according to the cloud account that is created, issue a command similar to this:

```
mmcloudgateway cloudStorageAccessPoint create --cloud-nodeclass TCTNodeClass1
                                   --cloud-storage-access-point-name AccessPoint1
                                   --account-name mycloud --url http://192.0.2.0
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmi.gpfs.net.
mmcloudgateway: Command completed.
```

- To create a CSAP with an https endpoint, issue a command similar to this:

```
mmcloudgateway cloudStorageAccessPoint create --cloud-nodeclass TCTNodeClass1
--cloud-storage-access-point-name AccessPoint1
--account-name mycloud --url https://192.0.2.0 --server-cert-path /root/ca.pem
```
- To update a CSAP with the slice size, issue a command similar to this:

```
mmcloudgateway cloudStorageAccessPoint update --cloud-nodeclass tct  --cloud-storage-access-point-name csap
--slice-size 524000
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with pk.slabs.ibm.com
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on pk.slabs.ibm.com.
mmcloudgateway: Command completed.
```
- To delete a CSAP, issue a command similar to this:

```
mmcloudgateway cloudStorageAccessPoint delete --cloud-nodeclass cloud
--cloud-storage-access-point-name csap1
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmip51.gpfs.net.
mmcloudgateway: Command completed.
```

Note: In proxy-based environments, set your proxy settings as part of the node class configuration before you run any migrations. If tiering commands (migrate/recall) are run before you set the proxy details, they might fail for not being able to reach out to the public cloud storage providers such as Amazon S3.

For more information, see the *mmcloudgateway command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Next step: See "Creating Cloud services."

Creating Cloud services

You must create at least one Cloud services for each cloud account that is created. You can associate a single Cloud services with only one cloud account but can use it with multiple file systems or file sets.

For data movement (sharing or tiering) commands, you must specify a Cloud service name to move data to the intended object storage if there is more than one Cloud service that is configured to the file system or file set. However, you do not have to specify a Cloud service name for these data movement commands if only one Cloud servicee is configured for a file system or file set.

Additionally, if you want to execute both tiering and sharing operations in your Cloud services setup, you must define one Cloud service for tiering and another for sharing.

- To create a Cloud service according to the cloud account that is created, issue a command similar to the following one:

```
mmcloudgateway cloudService create
                    --cloud-nodeclass TCTNodeClass1 --cloud-service-name mycloud
                    --cloud-service-type Tiering --account-name Cleversafe_cloud
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmi.gpfs.net.
mmcloudgateway: Command completed.
```

Note: You can use this Cloud service only for tiering. If you want to use it for sharing, you can replace *Tiering* with *Sharing*.

- To update Cloud services, issue a command according to the following:

```
mmcloudgateway cloudService update --cloud-nodeclass cloud --cloud-service-name newServ --disable
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmip51.gpfs.net.
mmcloudgateway: Command completed.
```

- To delete Cloud services, issue a command according to the following:

```
mmcloudgateway cloudService delete --cloud-nodeclass cloud --cloud-service-name newServ
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmip51.gpfs.net.
mmcloudgateway: Command completed.
```

Next step: "Configuring Cloud services with SKLM (optional)."

Configuring Cloud services with SKLM (optional)

In order to encrypt data being tiered to the cloud storage, you need to first configure a key manager, before creating container pair set in the next step. Two types of key manager are supported - local key manager (simple JCEKS based one) and IBM SKLM Server. You need to create one local key manager per cluster. The SKLM key manager is optional and might be created per cluster or per sets of file systems, depending on your security needs.

Before you configure Cloud services with IBM Security Key Lifecycle Manager, ensure that an SKLM server is installed. For more information, see the *Preparation for encryption* topic in the *IBM Spectrum Scale: Administration Guide*.

Note: Transparent cloud tiering only supports IBM Security Key Lifecycle Manager versions 2.6.0 and 2.7.0.

You can create a key manager if you want to use this parameter while you configure a container pair set in the next topic.

- To create an SKLM key manager, issue a command similar to the following command:

```
mmcloudgateway keymanager create --cloud-nodeclass cloud
                                 --key-manager-name vm1
                                 --key-manager-type RKM
                                 --sklm-hostname vm1
                                 --sklm-port 9080
                                 --sklm-adminuser SKLMAdmin
                                 --sklm-groupname tct
```

The system displays output similar to this:

```
Please enter a password:
Confirm your password:
mmcloudgateway: Sending the command to the first successful node starting with vmip51.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmip51.gpfs.net.
mmcloudgateway: Command completed.
```

- To rotate a key manager, issue a command according to the following:

```
mmcloudgateway keymanager rotate --cloud-nodeclass cloud --key-manager-name vmip131
```

The system displays output similar to the following:

```
mmcloudgateway: Sending the command to the first successful node starting with c01.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on c80f4m5n01.gpfs.net.
mmcloudgateway: Command completed.
```

- To update a key manager, issue a command according to the following:

```
mmcloudgateway keymanager update --cloud-nodeclass cloud
--key-manager-name sklm --update-certificate
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with c01.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on c01.gpfs.net.
mmcloudgateway: Command completed.
```

Next step: "Binding your file system or fileset to the Cloud service by creating a container pair set."

Note: The local key manager is simpler to configure and use. It might be your best option unless you are already using SKLM in your IBM Spectrum Scale cluster or in cases where you have special security requirements that require SKLM.

Binding your file system or fileset to the Cloud service by creating a container pair set

Up to this point, the configuration work was about creating access out to the cloud. Now, you need to bind all this to the data on your cluster, and you do that by creating a container pair set that associates your file system or fileset to the Cloud service. Once you create your container pair set, your Cloud service is usable.

Cloud data sharing internally creates two containers on cloud storage for storing data as well as meta-data. However, some cloud providers require containers to be created using its native interfaces. In that case, you need to provide the names. The containers that are created for Cloud data sharing can be shared with other file systems or Cloud services. However, the containers that are created for tiering cannot be shared. Creating the container pair set is how you bind a file system to a Cloud service.

You must decide whether or not to add encryption to the container at the time of creating it, and accordingly use the KeyManagerName parameter. If you create a container pair without a key manager, you will not be able to add encryption to the container at a later point by using the `mmcloudgateway ContainerPairSet update` command.

Note: In this release, containers do not have encryption enabled by default. Hence, administrators need to explicitly add "--enc ENABLE" parameter while creating a container pair set, to ensure that the data is encrypted while being tiered to a cloud storage.

You can create a container pair set with thumbnail enabled, and the scope can be enabled to either a file system or a fileset according to your business requirements.

Note:
- Changing the mount point for a file system or the junction path of a fileset after it is associated with a container is not supported.
- Container pair creation is not supported with an immutable fileset. To configure a container pair set using an immutable fileset with a fileset scope, you must provide a --cloud-directory-path (the database path) that points to any mutable fileset.

- You can enable or disable transparent recall policy by using the --transparent-recalls {ENABLE|DISABLE} parameter. However, this parameter is optional, and transparent recall policy is enabled by default even if you do not use this parameter.

Do the following steps for creating, testing, listing, or deleting a container pair set:

- To create a container pair set, issue a command similar to this:

```
mmcloudgateway containerpairset create --cloud-nodeclass TCTNodeClass1
                          --container-pair-set-name newContainer
                          --cloud-service-name myService
                          --scope-to-filesystem
                          --data-container data_container
                          --metadata-container meta_container
```

  The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with v1.gpfs.net
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on vmi.gpfs.net.
mmcloudgateway: Command completed.
```

  Note: If you do not specify the names for storing data and metadata containers, then the container pairset name is used for both data and metadata containers. In this example, they are "newContainer" (for data) and "newContainer.meta" (for metadata).

- To create a container pairset with thumbnail enabled and the scope is a file system, issue a command similar to this:

```
mmcloudgateway containerpairset create --cloud-nodeclass cloud --container-pair-set-name x13
--cloud-service-name newServ --scope-to-filesystem --thumbnail-size 64
```

- To create a container pairset when the scope is a fileset, issue a command similar to this:

```
mmcloudgateway containerpairset create --cloud-nodeclass cloud --container-pair-set-name x13
--cloud-service-name newServ --scope-to-fileset
```

- To create a container pair set that is enabled for encryption, issue a command similar to this:

```
mmcloudgateway containerpairset create --cloud-nodeclass tct --container-pair-set-name Containeretag5
--cloud-service-name csss5  --path /gpfs0/fs3 --enc ENABLE --etag ENABLE --data-container test5
--meta-container testmeta5 --key-manager-name lkm3 --scope-to-fileset
```

- To configure a container pair set using an immutable fileset with a fileset scope, issue a command similar to this:

```
mmcloudgateway containerpairset create --cloud-nodeclass tct --container-pair-set-name wormcp
--cloud-service-name wormservice2 --path /gpfs0/worm2 --enc ENABLE --etag ENABLE --data-container
wormtestnov --meta-container wormtestnovmeta --key-manager-name lkm3 --scope-to-fileset
--cloud-directory-path /gpfs0/fs3
```

  Note: Here, the fileset is an immutable fileset whereas the cloud directory is pointing to a fileset that is not immutable.

- To test a container pair set that is created, issue a command similar to this:

```
mmcloudgateway containerpairset test --cloud-nodeclass cloud --container-pair-set-name vmip51
```

- To delete a container pair set, issue a command similar to this:

```
mmcloudgateway containerpairset delete --container-pair-set-name x13
--cloud-nodeclass cloud
```

- To list a container pair set, issue a command similar to this:

```
mmcloudgateway containerpairset list
```

  The system displays output similar to this:

```
Configured 'containerPairSet' options from node class  cloud:
------------------------------------------------------------
    containerPairSetName         :  vmip51
    path                         :  /fs/
    scopeTo                      :  filesystem
```

```
|          transparentRecalls           : Enabled
|          cloudServiceName             : newServ
|          dataContainer                : vmip51
|          metaContainer                : vmip51.meta
|          thumbnailSize                : 0
|          cloudDirectoryPath           : /fs/
|          dataLocation                 :
|          metaLocation                 :
|          activeKey                    :
|          keyManagerName               :
|          containerPairSetName         : fset1
|          path                         : /fs/
|          scopeTo                      : filesystem
|          transparentRecalls           : Disabled
|          cloudServiceName             : newServ
|          dataContainer                : fset1
|          metaContainer                : fset1.meta
|          thumbnailSize                : 0
|          cloudDirectoryPath           : /fs/
|          dataLocation                 :
|          metaLocation                 :
|          activeKey                    :
|          keyManagerName               :
|
|   Configured 'containerPairSet' options from node class cloud2:
|   ---------------------------------------------------------
|          containerPairSetName         : vmip53
|          path                         : /x13/
|          scopeTo                      : filesystem
|          transparentRecalls           : Enabled
|          cloudServiceName             : serv2
|          dataContainer                : vmip53
|          metaContainer                : vmip53.meta
|          thumbnailSize                : 0
|          cloudDirectoryPath           : /x13/
|          dataLocation                 :
|          metaLocation                 :
|          activeKey                    :
|          keyManagerName               :
```

| **Note:** Now that you have created your container configuration, it is critical that you do the following:
| • Back up your configuration and security information for disaster recovery purposes. For more
| information, see "Backing up the Cloud services configuration" on page 64.
| • Regular maintenance on this container pair set. For more information, see "Scheduling
| recommendations for Transparent cloud tiering tasks" on page 672.

| For more information, see the *mmcloudgateway command* in the *IBM Spectrum Scale: Command and
| Programming Reference*.

| **Next step**: "Backing up the Cloud services configuration" on page 64

Backing up the Transparent cloud tiering database to the cloud

| Transparent cloud tiering database stores critical information, such as the list of files that are migrated to
| the cloud and the list of files that are deleted from the cloud, which is associated with each container. It
| is essential to back up this database for any type of disaster recovery.

| To back up the Transparent cloud tiering database, issue a command according to the following syntax:
| `mmcloudgateway files backupDB  --container-pair-set-name <coontainerpairsetname>`

| For example, to back up the database that is associated with the container, cpair1, issue this command:

```
| mmcloudgateway files backupDB --container-pair-set-name cpair1
```

| The system displays output similar to this:
```
| mmcloudgateway: Command completed.
```

| If the database size is large, then backing up operation can be a long running process.

| **Note:** By using the backed-up database, you can perform a database recovery by using the
| `mmcloudgateway files rebuildDB` command. For more information, see "Manual recovery of Transparent
| cloud tiering database" on page 668.

| Make sure you back up your Cloud data sharing database periodically, for example, once a month. For
| more information on scheduling maintenance activities, see "Scheduling recommendations for
| Transparent cloud tiering tasks" on page 672.

Backing up the Cloud services configuration

| It is critical that the Cloud services configuration data that is stored in the CCR is always backed up.

| To back up the configuration data, issue a command according to the following syntax:
```
| mmcloudgateway service backupConfig --backup-file <name of the file including the path>
```

| The following files are backed up from the CCR:
| • mmcloudgateway.conf
| • _tctkeystore.jceks
| • _nodegroup1.settings
| • _nodegroup2.settings

| **Note:** Files that are collected as part of backup are settings file for each node group.
| You can specify a path along with the file name. If the path does not exist, then the command creates the
| path. The backed-up files are stored in a tar file, which is saved under the specified folder. If the path is
| not specified, then the tar file is stored in the local directory.

| Refer to the following examples:
| • Issue the following command to back up the configuration data to the file, tct_config_backup under
| /tmp/mydir/, where the folder does not exist:
```
|   mmcloudgateway service backupconfig --backup-file /tmp/mydir/tct_config_backup
```

| The system displays output similar to this:
```
|   mmcloudgateway: Directory '/tmp/mydir' does not exist.  Creating it.
|   mmcloudgateway: Starting backup
|
|     Backup Config Files:
|       [mmcloudgateway.conf  -  Retrieved]
|       [_tctkeystore.jceks   -  Not Found]
|       [_cloudnodeclass.settings  -  Retrieved]
|       [_cloudnodeclass1.settings  -  Retrieved]
|
|   mmcloudgateway: Creating the backup tar file...
|   mmcloudgateway: Backup tar file complete.  The file is '/tmp/mydir/tct_config_backup_20170915_085741.tar
|   mmcloudgateway: The backup file should be archived in a safe location.
|   mmcloudgateway: Command completed.
```
| • Issue the following command to back up the configuration data to the file, tct_config_backup, under
| /tmp/mydir/, where the folder does exist:
```
|   mmcloudgateway service backupconfig --backup-file  /tmp/mydir/tct_config_backup
```

The system displays output similar to this:

```
mmcloudgateway: Starting backup

  Backup Config Files:
    [mmcloudgateway.conf  - Retrieved]
    [_tctkeystore.jceks   - Not Found]
    [_cloudnodeclass.settings  - Retrieved]
    [_cloudnodeclass1.settings  - Retrieved]

mmcloudgateway: Creating the backup tar file...
mmcloudgateway: Backup tar file complete.  The file is '/tmp/mydir/tct_config_backup_20170915_085741.tar
mmcloudgateway: The backup file should be archived in a safe location.
mmcloudgateway: Command completed.
```

In these examples, tct_config_backup is the name that is given to the tar file. The file name is appended with the date and time when the command is run. The format is *filename_yyyymmdd_hhmmss.tar*. By doing so, the file names are not overwritten even if an administrator runs this command multiple times, providing the same file name.

Note: It is a best practice to save the backup file in a safe location outside the cluster to ensure that the backup file can be retrieved even if the cluster goes down. For example, when you use encryption no copy of the key library is made to cloud storage by Transparent cloud tiering. Therefore, if there is a disaster in which a cluster is destroyed, you must make sure that the key library is safely stored on a remote cluster. Otherwise, you cannot restore the Transparent cloud tiering service for files that are encrypted on the cloud because the key to decrypt the data in the cloud is no longer available.

Enabling a policy for Cloud data sharing export service

This topic describes how to create a policy for export, a service provided by the IBM Spectrum Scale Cloud services, Cloud data sharing.

After you create a cloud storage account, you will want to create a policy for exporting data to cloud storage. A sample policy is provided below. You can run this policy manually as needed or set it up to run periodically – a cron job is frequently employed for this purpose. This policy is meant to run weekly and exports files greater than one day old and less than eight days old.

Note: Do not set this down to modified age of 0 if you want to run it in a cron job -- as it will only pick up a partial list of files for day 0 and you may end up with get gaps or duplicates over your week-to-week policy runs.

```
define(
modified_age,
(DAYS(CURRENT_TIMESTAMP) - DAYS(MODIFICATION_TIME))
)
RULE EXTERNAL LIST 'export' EXEC '/opt/ibm/MCStore/bin/mcstore' OPTS '-e'
RULE 'files-to-export'
LIST 'export'
WHERE modified_age > 1 AND modified_age <= 8
```

How this could be applied using a cron job:

1. Open a cron job editor by issuing this command:

   ```
   crontab -e
   ```

2. Add weekly export cron job by specifying this command:

   ```
   @weekly mmapplypolicy {Device|Directory} -P PolicyFile
   ```

3. Specify a command to export the journal to cloud storage here, if required.

4. Specify the asynchronous notification of new files in the cloud here (pick your favorite notification tool).

5. Save the file.

Tuning Cloud services parameters

This topic describes the tunable parameters for Cloud services and the commands to modify them.

Cloud services use some default configuration parameters. You can change the value of the parameters if the default settings do not suit your requirements.

The following table provides the list of configurable parameters and their description:

Table 5. Attributes and default values

Variable Name	Default Value	Minimum Value	Maximum Value	Description
connector. server.timeout	5000 (ms)	1000 (ms)	15000 (ms)	This is the maximum amount of time the server takes to respond to the client request. If the request is not fulfilled, it closes the connection.
connector. server.backlog	0	0	100	The maximum queue length for incoming connection indications (a request to connect) is set to the backlog parameter. If a connection indication arrives when the queue is full, the connection is refused.
destroy.sql.batchsize	8196	8196	81960	Page size per delete local database objects operation.
destroy. cloud.batchsize	256	256	81960	Page size per cloud objects delete operation.
reconcile.sql.batchsize	8196	8196	81960	Reconcile processes files in batches. This parameter controls how many files are processed in a batch.
commands. reconcile.lockwait.timeout.sec	360(s)	60(s)	3600(s)	Maximum time to acquire the lock on directory for the reconcile operation.
threads.gc.batchsize	4096	4096	40960	Page size of the Garbage Collector thread. We can increase this in case the memory usage is more.
migration. downgrade. lock.threshold.size.mb	64 (MB)	1 (MB)	64 (MB)	Sets the size threshold on files for which the lock downgrade is completed. To save time, a lock downgrade is not completed on shorter files that can transfer quickly. For larger files, a lock downgrade is suggested because migration might take a long time.
cloud-retention-period-days	30	0	2147483647	Number of days for which the migrated data needs to be retained on the cloud after its file system object has been deleted or reversioned.
connector. server.migrate-threadpool-size	32	1	64	Thread pool size of the migration threads. User can do this by making sure the CPU resources (CPU speed and number of cores) of the Cloud service nodes match.
connector.server.recall-threadpool-size	32	1	64	Thread pool size of recall threads. User can increase the number of recall threads to improve the performance.

Table 5. Attributes and default values (continued)

Variable Name	Default Value	Minimum Value	Maximum Value	Description
tracing.enable	true	true	False	Enables administrators to print trace messages of the internal components in a file. Controls the level of messages such as Info, Warning, or Error.
tracing.level	ALL=4	See Table 6	See Table 6	Tracing level is to set non-default tracing levels for various Transparent Cloud Tiering internal components to generate more debug data if any problems occur.
audit.enable	true	true	true	Enables or disables auditing information.
threads.cut-slow.sleep.ms	6000000 (ms)	60000 (ms)	2^63	Sleep time between two slow cloud update thread runs.
threads.cut-slow.sizediff	268435456 (Bytes)	1048576 (Bytes)	2^63	Size threshold of Cloud Updater Slow Threads.
threads.cut-slow.timediff.ms	604800000 (ms)	86400000 (ms)	2^63	Time threshold of Cloud Updater Slow Threads to update the cloud database.
threads.cut-fast.sizediff	16777216 (Bytes)	1048576 (Bytes)	2^63	Size threshold of Cloud Updater Fast Threads.
threads.cut-fast.timediff-active.ms	1800000 (ms)	60000 (ms)	2^63	Active time of Cloud Updater Fast Threads.

Table 6. Supported Components

Variable Name	Default Value
THRD	Threading, Thread Pools
STCK	The modular stack and how operations propagate up and down
BSCN	operations related to the blob store connection
STAT	Deals with States
MPAU	Multi-part upload
CNCT	Connectors
SLCE	Slices
KMGR	Key Management
ENCR	Encryption
GCON	Scale back end connector
ETAG	Etag Based Integrity
MFST	Manifest related Operations
PAYL	Payload related Operations
ENVR	Environment related operations
CDIR	Cloud Directory Component Operations
RECN	Reconcile
SCAN	Scan operations
POLI	GPFS Policy operations
AUTH	Authentication

Table 6. Supported Components (continued)

Variable Name	Default Value
SQLL	SQL operations
JRNL	Journal operations
NOTF	GPFS Event Notifications
MTRX	Metrics Code
CDAM	Core Data Model
GDAM	GPFS Data Model
MTTV	TCT TTV Validator
CONF	TCT configuration Layer related operations.
SERV	Serviceability
MMON	Monitoring

To tune Cloud services parameters, issue a command according to this syntax:

```
mmcloudgateway config set --cloud-nodeclass CloudNodeClass
                          Attributee=value[,Attributee=value...]
```

where,

- <--cloud-nodeclass> is the node class configured for the Cloud services nodes
- <Attribute> is the value of the attribute that is provided.

For more information, refer to the following examples:

- To set the value of the `tconnector.server.timeout` attribute to 10 seconds, issue this command:

  ```
  mmcloudgateway config set --cloud-nodeclass tct1 connector.server.timeout=10000
  ```

- To reset the value of the `tconnector.server.timeout` attribute to the default value, issue this command:

  ```
  mmcloudgateway config set --cloud-nodeclass tct1 connector.server.timeout=DEFAULT
  ```

- To set the tracing levels, issue this command:

  ```
  mmcloudgateway config set --cloud-nodeclass tct tracing.level=GCON=5:AUTH=4
  ```

- To reset the value of the tracing components to the default value, issue this command:

  ```
  mmcloudgateway config set --cloud-nodeclass tct tracing.level=default
  ```

- To reset the values of multiple attributes, issue this command:

  ```
  mmcloudgateway config set --cloud-nodeclass tct tracing.level=CDIR=1:
  JRNL=1:SQLL=1:RECN=1
  ```

Integrating Cloud services metrics with the performance monitoring tool

This topic describes the procedure for integrating Cloud services server nodes with the performance monitoring tool.

Performance monitoring collector (pmcollector) and sensor (pmsensors) services are installed under /opt/IBM/zimon.

Note: You must install the sensors on all the cloud service nodes, but you need to install the collectors on any one of the GPFS nodes. For installation instructions, see *Manually installing the Performance Monitoring tool* topic in the *IBM Spectrum Scale: Administration Guide*.

Two types of configurations are possible for the performance monitoring tool integration:

- GPFS-based configuration (configuration by using GPFS commands). This type is recommended.
- File-based configuration (manual configuration of the sensor files as described here)

For more information on each of these methods, see *Configuring the performance monitoring tool* in *IBM Spectrum Scale: Problem Determination Guide*.

GPFS-based configuration

This topic describes the procedure for integrating cloud service metrics with the performance monitoring tool by using GPFS-based configuration.

1. On the Cloud services nodes, copy the following files from `/opt/ibm/MCStore/config` folder to `/opt/IBM/zimon` folder:
   - TCTDebugDbStats
   - TCTDebugLweDestroyStats
   - TCTFsetGpfsConnectorStats
   - TCTFsetIcstoreStats
   - TCTFsGpfsConnectorStats
   - TCTFsIcstoreStats

2. Register the sensor in the GPFS configuration by storing the following snippet in the `MCStore-sensor-definition.cfg` file:

```
sensors=
{
    # Transparent cloud tiering statistics
    name = "TCTDebugDbStats"
    period = 10
    type = "Generic"
},

{
    #Transparent cloud tiering statistics
    name = "TCTDebugLweDestroyStats"
    period = 10
    type = "Generic"
},

{
    #Transparent cloud tiering statistics
    name = "TCTFsetGpfsConnectorStats"
    period = 10
    type = "Generic"
}
{
    #Transparent cloud tiering statistics
    name = "TCTFsetIcstoreStats"
    period = 10
    type = "Generic"
},

{
    #Transparent cloud tiering statistics
    name = "TCTFsGpfsConnectorStats"
    period = 10
    type = "Generic"
},

{
    #Transparent cloud tiering statistics
```

```
|                name = "TCTFsIcstoreStats"
|                period = 10
|                type = "Generic"
|            }
```
   3. Run this command:
```
      prompt# mmperfmon config add --sensors MCStore-sensor-definition.cfg
```

   Note: The sensor definition file can list multiple sensors separated by commas (,).

For more information on GPFS-based configuration, see the topic *mmperfmon command* in the *IBM Spectrum Scale: Command and Programming Reference* guide.

File-based configuration

This topic describes how to configure Cloud services with the performance monitoring tool by using file-based (manual) configurations.

Note: You must delete the sensors that are used in the earlier releases. If the scope of your Cloud services configuration is file system, then you do not need to configure the sensor files that start with *TCTFset**. Similarly, if the scope of your Cloud services configuration is fileset, then you do not need to configure the sensor files that start with *TCTFs**.

To integrate the performance monitoring tool with Cloud services server nodes, do the following steps:

   1. Copy /opt/IBM/zimon/defaults/ZIMonSensors.cfg to /opt/IBM/zimon. This configuration file determines which sensors are active and their properties.

   Note: If the sensors are already configured at /opt/IBM/zimon/defaults/ZIMonSensors.cfg, use the same sensors.

   2. Edit the /opt/IBM/zimon/ZIMonSensors.cfg file and set an IP address for the "host" attribute in the "collectors" section.

   Note: If the collectors are already configured at /opt/IBM/zimon/ZIMonSensors.cfg, use the same collectors.

   3. Edit the /opt/IBM/zimon/ZIMonSensors.cfg file to append the following sensors at the end of the sensor configuration section:
```
|    sensors=
|    {
|        # Transparent cloud tiering statistics
|        name = "TCTDebugDbStats"
|        period = 10
|        type = "Generic"
|      },
|
|        {
|         #Transparent cloud tiering statistics
|         name = "TCTDebugLweDestroyStats"
|         period = 10
|         type = "Generic"
|       },
|
|      {
|        #Transparent cloud tiering statistics
|        name = "TCTFsetGpfsConnectorStats"
|        period = 10
|        type = "Generic"
|      }
|    {
|        #Transparent cloud tiering statistics
|        name = "TCTFsetIcstoreStats"
|        period = 10
|        type = "Generic"
```

```
      },

    {
        #Transparent cloud tiering statistics
        name = "TCTFsGpfsConnectorStats"
        period = 10
        type = "Generic"
      },

    {
        #Transparent cloud tiering statistics
        name = "TCTFsIcstoreStats"
        period = 10
        type = "Generic"
      }
```

> **Note:** Each sensor should be separated by a comma. The period is the frequency in seconds at which the performance monitoring tool polls the cloud service for statistics. The period is set to 10 seconds but it is a configurable value. The sensor is turned off when the period is set to 0.

4. Copy the following files from /opt/ibm/MCStore/config folder to /opt/IBM/zimon folder.
   - TCTFsGpfsConnectorStats.cfg
   - TCTFsIcstoreStats.cfg
   - TCTFsetGpfsConnectorStats.cfg
   - TCTFsetIcstoreStats.cfg
   - TCTDebugLweDestroyStats.cfg
   - TCTDebugDbStats.cfg

5. Restart the sensors by using this command: **service pmsensors restart** .
6. Restart the collectors by using these commands: **service pmcollector restart**

   Note:

   If the collector is already installed and is running, then only **pmsensors** service needs to be restarted. If you are installing both **pmcollectors** and **pmsensors**, then both services need to be restarted.

Setting up Transparent cloud tiering service on a remotely mounted client

When you use Transparent cloud tiering on a remote cluster, you must complete your administrative work and policy migration scheduling on the Transparent cloud tiering server nodes. User access to tiered data from remotely mounted clusters is available by setting up transparent recalls (which is outlined here).
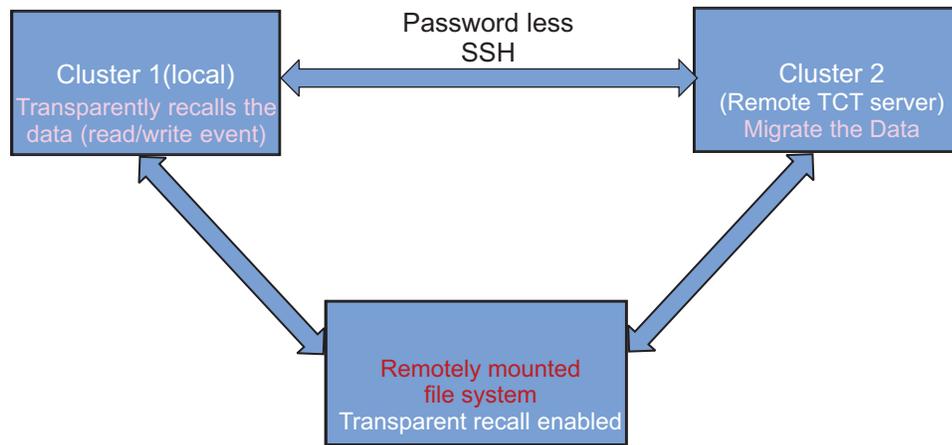
Figure 2. Transparent cloud tiering on a remote cluster

Before you begin, ensure the following:

- A local cluster and a remote cluster are created. For more information, see the *mmcrfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.
- A passwordless SSH is set up between the local and the remote clusters. For instructions on passwordless SSH setup and other prerequisites, see the *Prompt-less SSH setup* section of the *Problems due to missing prerequisites* topic in the *IBM Spectrum Scale: Problem Determination Guide*.
- Transparent cloud tiering server RPMs are installed on the local cluster.

1. Create authentication between the local cluster and the remote cluster. For more information, see the *Mounting a remote GPFS file system* topic in the *IBM Spectrum Scale: Administration Guide*.

2. Verify that the file system is mounted on the remote cluster (from the local cluster) by issuing the following command on the remote cluster:

   ```
   mmremotecluster show
   ```

   The system displays output similar to this:

   ```
   Cluster name:    vm1.pk.slabs.ibm.com
   Contact nodes:   vm1.pk.slabs.ibm.com
   SHA digest:      37a856428f565d017gg4abb935a81493d66cd0498917e8ef750c1b5e4bc60d56
   SHA digest:      mygpfsnew (gpfs0)
   ```

3. On the local cluster, set the following GPFS variable, which will point to the gateway node of the remote cluster that needs to be connected for transparent recall:

   a. For a single remote cluster mounted, set the variable, as follows:

   ```
   mmccr vput tct<remote_cluster_name> <IP_address>
   ```

   For example, to set the GPFS variable on local cluster for remote cluster tctvm1.pk.slabs.ibm.com, issue the following command:

   ```
   mmccr vput tctvm1.pk.slabs.ibm.com 192.0.2.0
   ```

   b. For multiple remote cluster mounted, set the variable, as follows:

   ```
   mmccr vput tct<remote_cluster1_name> <IP_address>
   mmccr vput tct<remote_cluster2_name> <IP_address>
   ```

   For example, to set the GPFS variable on local cluster for two remote clusters, tctvm1.pk.slabs.ibm.com and tctvm2.pk.slabs.ibm.com, issue the following command:

   ```
   mmccr vput tctvm1.pk.slabs.ibm.com 198.51.100.1
   mmccr vput tctvm2.pk.slabs.ibm.com 198.51.100.2
   ```

4. Copy the mcstore script from the local cluster to the remote cluster by issuing the following command:

```
cp /opt/ibm/MCStore/scripts /opt/ibm/MCStore/bin/
```

5. Migrate data from the local cluster to the cloud by using the **mmcloudgateway files migrate** command.

6. Transparently recall data from the cloud (by issuing the **cat** or any similar command on the remote cluster CLI).

Deploying WORM solutions

This topic describes how you can set up a WORM (write once and read many) solution by using IBM Spectrum Scale, Transparent cloud tiering, and IBM Cloud Object Storage.

IBM Spectrum Scale provides the immutability feature where you can associate a retention time with files, and any change or deletion of file data is prevented during the retention time. You can configure a IBM Spectrum Scale fileset with an integrated Archive Manager (IAM) mode by using the **mmchfileset** command. Files stored in such an immutable fileset can be set to immutable or append-only by using standard POSIX or IBM Spectrum Scale commands. For more information on immutability features available in IBM Spectrum Scale, see "Immutability and appendOnly features" on page 408.

After immutability feature is configured in IBM Spectrum Scale, you can ensure that files that are stored on the Object Storage are immutable by leveraging the locked vault feature available in IBM Cloud Object Storage.

Locked vaults enable storage vaults to be created and registered under the exclusive control of an external gateway application. IBM Cloud Object Storage stores objects received from the gateway application. The gateway authenticates to the IBM Cloud Object Storage Manager exclusively by using an RSA private key and certificate that was configured to create a locked vault and registered only with the gateway. After that, the normal S3 APIs can be used against the Accesser® nodes by using the configured private key and certificate. Accesser API key and secret key for S3 API cannot be used for authentication or authorization. If a key is compromised, the gateway rotates keys by calling the Rotate Client Key Manager REST API. This API replaces the existing key and revokes the old certificates. A locked vault with data cannot be deleted by the IBM Cloud Object Storage Administrator, and its ACLs cannot be changed. Additionally, it cannot be renamed or have proxy setting enabled. For more information about locked vaults, see *IBM Cloud Object Storage System Locked Vault Guide*.

Note: To configure WORM feature at the fileset level, it is recommended to match the immutable filesets with immutable container pair sets on the cloud.

Creating immutable filesets and files

In order to configure and deploy a WORM solution, it is mandatory to create an immutable fileset in IBM Spectrum Scale.

1. Create an independent fileset by using the **mmcrfileset** command.

2. Link the fileset to a directory within the file system which must not exist at this point:

   ```
   mmlinkfileset <file system name> <fileset name> -J directory
   ```

   Note: This directory is the immutable fileset path.

3. Set an IAM mode for the files by using the following command:

   ```
   mmchfileset <file system name> <fileset name> --iam-mode compliant
   ```

4. Create a test file `date > testfile` with read-write permissions and fill the file with some content:

   ```
   echo "Hello World" > file
   ```

5. Check the extended attributes of the file which indicate that the file is not immutable by using the **mmlsattr** command:

```
[root@localhost WORMfs]# mmlsattr -L testfile
file name: testfile
metadata replication: 1 max 2
data replication: 1 max 2
immutable: no
appendOnly: no
indefiniteRetention: no
expiration Time: Wed Mar 15 17:16:13 2016
flags:
storage pool name: system
fileset name: WORMfs
snapshot name:
creation time: Wed Mar 15 17:16:13 2016
Misc attributes: ARCHIVE
Encrypted: no
```

6. Set the file to read-only:

   ```
   chmod –w testfile
   ```

7. Set the future expiration time using **mmchattr**. Select a time in the immediate future for quick expiry and deletion.

   ```
   mmchattr --expiration-time 2016-03-15@18:16:13 testfile
   ```

8. Verify that immutability and expiration time are set by using **mmchattr**:

   ```
   mmlsattr -L testfile
   file name: testfile
   metadata replication: 1 max 2
   data replication: 1 max 2
   immutable: yes
   appendOnly: no
   indefiniteRetention: no
   expiration Time: Wed Mar 15 18:16:13 2016
   flags:
   storage pool name: system
   fileset name: WORMfs
   snapshot name:
   creation time: Wed Mar 15 17:16:13 2016
   Misc attributes: ARCHIVE READONLY
   Encrypted: no
   ```

9. Verify that the files cannot be modified or deleted. Run the following commands:

   ```
   # chmod +w testfile
   ```

   The system displays an output similar to this:

   ```
   chmod: changing permissions of 'testfile': Operation not permitted
   ```

   ```
   # date > testfile
   ```

   The system displays an output similar to this:

   ```
   testfile: Read-only file system
   ```

   ```
   # rm -f testfile
   ```

   The system displays an output similar to this:

   ```
    rm: cannot remove 'testfile': Read-only file system
   ```

   For more information, see the Immutability and appendOnly features in *IBM Spectrum Scale: Administration Guide*.

Setting up Transparent cloud tiering for WORM solutions

Once an immutable fileset is configured, you must set up Transparent cloud tiering to be able to configure and deploy WORM solutions. This topic describes the procedure for doing that.

Before you begin, ensure the following:

- IBM Cloud Object Storage version is at 3.9.x or later.
- IBM Cloud Object Storage is run on the "vault" mode, not on the "container" mode.
- To run the scripts, you have to create a private account on the IBM Cloud Object Storage.

Note: You can perform these procedures either manually or by using the scripts available in the package.

The following scripts are available at `/opt/ibm/MCStore/scripts`:
- `mcstore_createlockedvault.sh`
- `mcstore_lockedvaultpreconfig.sh`
- `mcstore_lockedvaultrotateclientkey.sh`
1. Set up a private key and create locked vaults
2. Configure Transparent cloud tiering with certificate-based authentication and locked vaults.
3. Rotate client key or revoke old certificate.
4. Update Transparent cloud tiering with new private key and certificate.

Setting up a private key and creating locked vaults

The first step involves creating a certificate signing request (CSR), registering certificate with IBM Cloud Object Storage via Client Registration REST API and obtaining a private key (RSA based).

Once the private key is obtained, it can be used to create the locked vaults on the IBM Cloud Object Storage system. Additionally, for HTTPS (TLS), the CA certificate of the IBM Cloud Object Storage system is also required.

The two locked vaults required for Transparent cloud tiering (data and metadata vaults) need to be created on the IBM Cloud Object Storage by using the *create vault from template* REST API. Once these vaults are created, they can be specified on the **mmcloudgateway filesystem create** command via the `—container-prefix` option.

Setting up a private key and a private certificate:

This topic describes the procedure for setting up a private key and a private certificate for deploying WORM solutions by using IBM Cloud Object Storage.

The first step involves creating a certificate signing request (CSR) and registering the client certificate with IBM Cloud Object Storage Manager via Client Registration REST API and obtaining a private key (RSA based) signed with IBM Cloud Object Storage Manager Certificate Authority. Once the signed private certificate is obtained, we can use the RSA private key and private certificate for creating the locked vaults on the IBM Cloud Object Storage system. Additionally, for HTTPS (TLS) communication, the root CA certificate of the IBM Cloud Object Storage system is also required.

Note:
- A private account must be created before an automation script or procedure is run.
- A private account must be created each time an incorrect IBM Cloud Object Storage CA certificate is specified while generating the Keystore.
1. Create a directory that will hold private key and certificates by issuing this command:
   ```
   $mkdir mydomain2.com.ssl/
   ```
2. To generate a keystore that will store the private key, CSR, and certificates, issue the following command in the `/opt/ibm/MCStore/jre/bin` directory:
   ```
   keytool -genkey -alias mydomain2 -keyalg RSA -keysize 2048 -keystore mydomain2.jks
   ```

   Note: You should make a note of the alias name as it has to be used in the later steps.
3. Generate CSR by issuing the following command:
   ```
   keytool -certreq -alias mydomain2 -keyalg RSA -file mydomain2.csr -keystore mydomain2.jks
   ```

4. Create a private account on IBM Cloud Object Storage Manager.
5. Using the private account created, send the CSR to IBM Cloud Object Storage Manager to be signed by issuing the following command:

```
curl -u <privateuser>:<password>
-k 'https://<COS Manager IP>/manager/api/json/1.0/clientRegistration.adm'
-d 'expirationDate=1508869800000' --data-urlencode 'csr=-----BEGIN NEW CERTIFICATE REQUEST-----

MIICzjCCAbYCAQAwWTELMAkGA1UEBhMCSU4xCzAJBgNVBAgTAktBMRIwEAYDVQQHEwlCYW5nYWxv
  cmUxDDAKBgNVBAoTA1NEUzENMAsGA1UECxMESVNETDEMMAoGA1UEAxMDSUJNMIIBIjANBgkqhkiG
  9w0BAQEFAAOCAQ8AMIIBCgKCAQEApfVgjnp9vBwGA6Y/g54DBr1wWtWeSAwm680M42O1PUuRwV92
  9UDBK9XEkY2Zb+o08Hvspd5VMU97bV7cnN8Fi8WuujHCdgAVuezTT0ZCHjVHl2L6CYql7hmWIazk
  TOaROoYlhzZCgQrDyVNIw6XuvkWo3eUIRyi1r6nafUFiqUtMEerEhEYa6cmm5qpeb2GKYJdeN53W
  SF0yrUCi9gRgPJiAq6lVSl+wWekbI6lwIAtJVyojx93lRl/KdxfFmh/sriUx//a6+I0OBli6EmEV
  BsHeG2HccS1diJ4+eUetXvfkYMjO6kRvYraSVKX022a4Jqki8iYDNf4XvRzOz5YbLQIDAQABoDAw
  LgYJKoZIhvcNAQkOMSEwHzAdBgNVHQ4EFgQUrgpT7F8Z+bA9qDxqU8PDg70zFj4wDQYJKoZIhvcN
  AQELBQADggEBADW4xuxBaaH9/ZBLOll0tXveSHF8Q4oZo2MhSWf34Shu/ZxC17H8NqCCMyxqVdXI
  6kbdg1se5WLCq/JJA7TBcgCyJJqVjADt+RC+TGNc0NlsC7XpeRYLJtxqlKilsWnKJf5oRvA1Vg5P
  nkTjCE9XvUzhJ/tTQjNBJSh8nN7Tbu/q5mTIGG9imARPro2xQpvwiFMHrq/f1uNeZ3SeuLxwQtkK
  4zge7XwyY63lrKsN0z2a4CPNzU0q68TGL1aE93QDpJYusSeTB0m2om4iTSNgsQKRmYqGDSXM3no/
  90UeTAgHjhJ82bGEOfP9FVm+6FnYydr1Endg1aEizC+sArk4e8E=
  -----END NEW CERTIFICATE REQUEST-----' -v
```

   Note: The expiration time should be specified in milliseconds.
6. Curl command provides a certificate in the response, as follows:

```
"-----BEGIN CERTIFICATE-----
\nMIIEczCCAlugAwIBAgIQeijQBskfm0v3kYQcBOBmxTANBgkqhkiG9w0BAQ0FADCB\nkTELMAkGA1UE
BhMCVVMxETAPBgNVBAgMCElsbGlub2lzMRAwDgYDVQQHDAdDaGlj\nYWdvMRMwEQYDVQQKDApDbGV2ZX
JzYWZlMRkwFwYDVQQDDBBkc05ldCBNYW5hZ2Vy\nIENBMS0wKwYDVQQFEyQwMmQxMjk5ZS05Nzc3LTRl
NmItODg3Yy0wYmMzNzJkODU1\nMzcwHhcNMTYxMDI0MTMxNTE2WhcNMTcxMDI0MTgzMDAwWjBZMQswCQ
YDVQQGEwJJ\nTjELMAkGA1UECBMCS0ExEjAQBgNVBAcTCUJhbmdhbG9yZTEMMAoGA1UEChMDU0RT\nMQ
0wCwYDVQQLEwRJU0RMMQwwCgYDVQQDEwNJQk0wggEiMA0GCSqGSIb3DQEBAQUA\nA4IBDwAwggEKAoIB
AQCl9WCOen28HAYDpj+DngMGvXBa1Z5IDCbrzQzjY7U9S5HB\nX3b1QMEr1cSRjZlv6jTwe+yl3lUxT3
ttXtyc3wWLxa66McJ2ABW57NNPRkIeNUeX\nYvoJiqXuGZYhrORM5pE6hiWHNkKBCsPJU0jDpe6+Rajd
5QhHKLWvqdp9QWKpS0wR\n6sSERhrpyabmql5vYYpgl143ndZIXTKtQKL2BGA8mICrqVVKX7BZ6RsjqX
AgC0lX\nKiPH3eVGX8p3F8WaH+yuJTH/9rr4jQ4GWLdxxLV2Inj55R61e9+Rg\nyM7qRG
9itpJUpfTbZrgmqSLyJgM1/he9HM7PlhstAgMBAAEwDQYJKoZIh*    vcNAQEN\nBQADggIBAJmCnhIN/
nhp2VIgqA7td3EBD8xrejF0bT5mSUgx8flFmCKCJh6/0yn9\nl1PUp3SzSu734GdTDZiUtTXax7PYZlB
3STlY0sZE7yU6zal0lIoUZEzXoohIEPVU\nW4X3j9HF3hWDwNsuqZfQDRmndaz6NG2EPDxiWgTYXPLdY
aZyTQFFe6A4tbT9gSHu\n9UD1woFwjrSAfg03zwR7wSRSwcALsVs1BK96TYufZf+E2eFg+QBGAC5YWrZ
i3g4Q\n1Xqxj5W5TwujLxSJ+8zxf6P9f0T96vGICH8Yy9AIWzUa3fXLh6tc1Pw+LbuIjEWr\nK2TS+DL
TmBAo8pQ5GsR8rShKFcPYOho2mbskAKgt4n+s63Jhu5qALS4Lw7eEQ7W7\nqGffZ2JttNHwePAAqvx33
xf+Y2SWn0fbOAlwT9BQ6ySn/qZR3e3Xl0rVqqukgCqO\nBnQhI5WN4HkONkyaquJruTLHUlWX5T01q/y
LnrRt8TCBA4qnX7HMlEmQkXiF5Poj\nBcyCTctYu1HlijHjsWO9kztUfljI5OkVyS1q1FqcZQiziHHRi
AEWbnrYn6Fgq13g\nIws7Lw9Utogj54tPCwJ8gEkoW4eTO4tnZmPTTdWlmVhTdEjVRxE8fotztHJuVis
P\nmFCxBPWJZ8IP9t2C/4Zi1PuqXI/8YZx8LPIcQUcRxeLURIgQrpb7
\n-----END CERTIFICATE-----\n"
```

7. Remove the '/n' character from the certificate (from BEGIN to END CERTIFICATE) and store the certificate in a file.
8. Get the CA certificate of IBM Cloud Object Storage Manager and import into the keystore created in step 2. To import the CA certificate, issue the following command:

```
keytool -importcert -trustcacerts -noprompt -alias cleversafeca -file
<cleversafe-cafile-loc>  -keystore mydomain2.jks -storepass  <keystore-password>
```

9. Import the certificate into the keystore by issuing the following command:

```
keytool -importcert -trustcacerts -alias mydomain2 -file <client-cert-location> -keystore
mydomain2.jks -storepass  <keystore-password>
```

   Note: You can set up a private key and a private certificate by using this script
   mcstore_lockedvaultpreconfig.sh available at /opt/ibm/MCStore/scripts, as follows:
   Setting up a private key and private certificate by using the automation script

a. Run **mcstore_lockedvaultpreconfig.sh** <keystorealiasname> <keycertLocationDirectory> <COSManagerIP> <username> <expirationDays> <COSCACertFile>, where the first 4 arguments are mandatory and the last two (expirationDays and COSCACertFile) are optional.

If the expiration date (expirationDays) is not specified, then the command will take the default expiration time, which is 365 days.

If the IBM Cloud Object Storage CA certificate (COSCACertFile) is not specified, then the CA file will be downloaded from the IBM Cloud Object Storage Manager.

b. For more information on the description of the parameters, see the **mmcloudgateway** man page.

For example,

```
./mcstore_lockedvaultpreconfig.sh test /root/svt 9.10.0.10 newuser2
```

The system displays output similar to this:

```
Enter KeyStore Password:
Enter Private Account Password:
Validating the inputs and the configuration....
COS Manager is reachable. Proceeding with Configuration...

Transparent Cloud Tiering Server RPM already installed. Proceeding with Configuration...

Python libraries are already installed. Proceeding with Configuration...

CURL already installed. Proceeding with Configuration...

Downloading COS CA Certificate....
Validation completed for inputs and the proceeding with configuration....
Generating a new Keystore and Private Key...
What is your first and last name?
  [Unknown]:  dmeo1
What is the name of your organizational unit?
  [Unknown]:  dmeo1
What is the name of your organization?
  [Unknown]:  demo2
What is the name of your City or Locality?
  [Unknown]:  demo1
What is the name of your State or Province?
  [Unknown]:  demo
What is the two-letter country code for this unit?
  [Unknown]:  KA
Is CN=dmeo1, OU=dmeo1, O=demo2, L=demo1, ST=demo, C=KA correct? (type "yes" or "no")
  [no]:  yes

Importing COS CA Certificate to Key Store.....
Certificate was added to keystore
Generating a CSR....
Sending CSR to CleverSafe to be signed.....
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2990  100  1781  100  1209   5310   3605 --:--:-- --:--:-- --:--:--  5316
Retrieving Certificate from Response.....
Importing Client Certificate to Keystore.....
Certificate reply was installed in keystore
Pre-configuration for Locked Vault completed successfully.
IMPORTANT: /root/svt/test.ssl contains private key, keystore and private certificate.
You must keep a back up of /root/svt/test.ssl.
```

Creating locked vaults:

Two locked vaults are to be created for deploying WORM solutions by using IBM Spectrum Scale.

IBM Cloud Object Storage Manager enables administrators to create vaults, which are under the exclusive control of a given external application (Transparent cloud tiering). This allows the application to have full control over the vault, but does not allow a user or administrator to bypass the application and directly

access the vault. Users are allowed to create WORM-style vaults that enforce read or write restrictions on the objects in the vault, which an administrator cannot bypass.

The two locked vaults required for Transparent cloud tiering (data and metadata vaults) need to be created on the IBM Cloud Object Storage by using Create vault from the template REST API. Once these vaults are created, they can be specified on the `mmcloudgateway filesystem create` command via the –container-prefix option.

Note: You can create a locked vault by using the `mcstore_createlockedvault.sh` script available at `/opt/ibm/MCStore/scripts`.

1. Convert the JKS keystore to the PKCS12 format by issuing this command:

   ```
   keytool -importkeystore -srckeystore mydomain2.jks -destkeystore new-store.p12 -deststoretype PKCS12
   ```

2. Extract the private key and convert it to an RSA key by issuing the following commands:
   - ```
     openssl pkcs12 -in "<keystore_directory>"/newkeystore.p12 -nocerts
     -out "<keystore_directory>"/privateKey.pem -passin pass:<keystore_password>
     -passout pass:<keystore_password>
     ```
   - ```
     openssl rsa -in "<keystore_directory>"/privateKey.pem -out "<keystore_directory>"/rsaprivateKey.pem
     -passin pass:<keystore_password>
     ```

3. By using the private key and certificate, create a locked vault (one for data and one for metadata) by issuing the following commands:
   - For data vault:

     ```
     curl --key ./ privateKeynew.pem --cert  <certificate-file>  -k -v
     'https://9.114.98.187/manager/api/json/1.0/createVaultFromTemplate.adm'
     -d 'id=1&name=demolockedvault&description=newlockedvaultdescription'
     ```
   - For metadata vault:

     ```
     curl --key ./ privateKeynew.pem --cert <certificate-file>  -k -v
     'https://9.114.98.187/manager/api/json/1.0/createVaultFromTemplate.adm'
     -d 'id=1&name=demolockedvault.meta&description=newlockedvaultmetadescription'
     ```

   Note: To find the provisioning template IDs, on the IBM Cloud Object Storage Manager GUI, click **Template Management**, hover the mouse over the template that is listed under **Vault Template**, and find the number that is displayed on the footer.

4. Print the locked vaults by issuing this command:

   ```
   curl --key privateKeynew.pem --cert <certificate-file>  -k '<COS Accesser IP Address>'
   ```

   Note: The names of the locked vaults must be noted down, and they must be specified to the `mmcloudgateway filesystem create` command by using the --container-prefix option.
   Creating a locked vault by using automation scripts

   a. Go to `/opt/ibm/MCStore/scripts` and run **mcstore_createlockedvault.sh** <keystorealiasname> <keyStorePath> <lockeddatavaultname> <lockeddatavaultDescription> <lockedmetavaultname> <lockedmetavaultDescription> <COSManagerIP> <dataVaultTemplateID> <metaVaultTemplateID>, where all parameters are mandatory.

   b. For description of the parameters, see the **mmcloudgateway** command.

      For example, **mcstore_createlockedvault.sh test /root/svt/test.ssl/test.jks demodatacontainer test demometacontainer metacontainer 9.10.0.10 1 1**.

      The system displays output similar to this:

      ```
      Enter KeyStore Password:
      Validating the inputs and the configuration....
      COS Manager is reachable. Proceeding with Configuration...

      Transparent Cloud Tiering Server RPM already installed. Proceeding with Configuration...
      openssl libraries are already installed. Proceeding with Configuration...

      curl already installed. Proceeding with Configuration...
      Certificate stored in file </root/svt/test.ssl/test_new.crt>
      ```

```
|          Creating locked vault...
|          MAC verified OK
|          writing RSA key
|          Locked data vault creation completed successfully.
|          Creating locked meta vault demofeb15.meta
|
|          Creating of Data and Meta Locked Meta Vault completed successfully.
|          Use mmcloudgateway filesystem create command to configure transparent cloud tiering
|          with locked vault.
```

Configuring Transparent cloud tiering with certificate-based authentication and locked vaults

This topic provides how to configure Transparent cloud tiering with certificate-based authentication and locked vaults.

Be sure to keep a backup copy of the source keystore that you used to import the private key and certificates. The **mmcloudgateway account delete** command removes the private key and certificates from the trust store.

1. Get the client certificate for IBM Cloud Object Storage Accesser.
2. Create a cloud storage account by using the **mmcloudgateway account create** command. For more information, see "Managing a cloud storage account" on page 57.
3. Create a cloud storage access point (CSAP) by using the **mmcloudgateway containerPairSet create** command. For more information, see "Binding your file system or fileset to the Cloud service by creating a container pair set" on page 61.
4. Create a cloud service by using the **mmcloudgateway cloudservice create** command. For more information, see "Creating Cloud services" on page 59.
5. Configure Cloud services with SKLM by using the **mmcloudgateway keyManager create** command. For more information, see "Configuring Cloud services with SKLM (optional)" on page 60.
6. Create a container pair set by using the **mmcloudgateway containerpairset create** command. For more information, see "Binding your file system or fileset to the Cloud service by creating a container pair set" on page 61.
7. Perform migrate and recall operations by using commands or policies.

Rotating Client Key or revoking old certificate

Once the client key is rotated, you must use the new certificate and private key to be able to create locked vaults. You can perform this procedure by using the following steps or by using this script: /opt/ibm/MCStore/scripts/mcstore_lockedvaultrotateclientkey.sh.

Note: Before you perform this procedure, ensure that no active migration is currently taking place. After you perform this procedure, the old keys will not work.

1. Generate a new CSR using a new alias:

   ```
   keytool -certreq -alias mydomainnew -keyalg RSA -file mydomainnew.csr -keystore mydomain2.jks
   ```

2. Get the CSR signed by sending it to the IBM Cloud Object Storage Manager:

   ```
   curl --cacert {path to ca certificate} --key {path to RSA private key}
   --cert {path to old certificate}
   'https://<COS Manager IP>/manager/api/json/1.0/rotateClientKey.adm'
   -d 'expirationDate=1508869800000' --data-urlencode 'csr=
   -----BEGIN NEW CERTIFICATE REQUEST-----
     MIICzjCCAbYCAQAwWTELMAkGA1UEBhMCSU4xCzAJBgNVBAgTAktBMRIwEAYDVQQHEwlCYW5nYWxv
     cmUxDDAKBgNVBAoTA1NEUzENMAsGA1UECxMESVNETDEMMAoGA1UEAxMDSUJNMIIBIjANBgkqhkiG
     9w0BAQEFAAOCAQ8AMIIBCgKCAQEApfVgjnp9vBwGA6Y/g54DBr1wWtWeSAwm680M42O1PUuRwV92
     9UDBK9XEkY2Zb+o08Hvspd5VMU97bV7cnN8Fi8WuujHCdgAVuezTT0ZCHjVHl2L6CYql7hmWIazk
     TOaROoYlhzZCgQrDyVNIw6XuvkWo3eUIRyi1r6nafUFiqUtMEerEhEYa6cmm5qpeb2GKYJdeN53W
     SF0yrUCi9gRgPJiAq6lVSl+wWekbI6lwIAtJVyojx93lRl/KdxfFmh/sriUx//a6+I0OBli6EmEV
     BsHeG2HccS1diJ4+eUetXvfkYMjO6kRvYraSVKX022a4Jqki8iYDNf4XvRzOz5YbLQIDAQABoDAw
     LgYJKoZIhvcNAQkOMSEwHzAdBgNVHQ4EFgQUrgpT7F8Z+bA9qDxqU8PDg70zFj4wDQYJKoZIhvcN
     AQELBQADggEBADW4xuxBaaH9/ZBLOll0tXveSHF8Q4oZo2MhSWf34Shu/ZxC17H8NqCCMyxqVdXI
   ```

```
      6kbdg1se5WLCq/JJA7TBcgCyJJqVjADt+RC+TGNc0NlsC7XpeRYLJtxqlKilsWnKJf5oRvA1Vg5P
      nkTjCE9XvUzhJ/tTQjNBJSh8nN7Tbu/q5mTIGG9imARPro2xQpvwiFMHrq/f1uNeZ3SeuLxwQtkK
      4zge7XwyY63lrKsN0z2a4CPNzU0q68TGL1aE93QDpJYusSeTB0m2om4iTSNgsQKRmYqGDSXM3no/
      90UeTAgHjhJ82bGEOfP9FVm+6FnYydr1Endg1aEizC+sArk4e8E=
      -----END NEW CERTIFICATE REQUEST-----' -v
```

3. Curl command provides a new certificate, as follows:

```
"-----BEGIN CERTIFICATE-----
\nMIIEczCCAlugAwIBAgIQeijQBskfm0v3kYQcBOBmxTANBgkqhkiG9w0BAQ0FADCB
\nkTELMAkGA1UEBhMCVVMxETAPBgNVBAgMCElsbGlub2l6MRAwDgYDVQQHDAdDaGlj
\nYWdvMRMwEQYDVQQKDApDbGV2ZXJzYWZlMRkwFwYDVQQDDBBkc05ldCBNYW5hZ2Vy
\nIENBMS0wKwYDVQQFEyQwMmQxMjk5ZS05Nzc3LTRlNmItODg3Yy0wYmMzNzJkODU1
\nMzcwHHcNMTYxMDI0MTMxNTE2WhcNMTcxMDI0MTgzMDAwWjBZMQswCQYDVQQGEwJJ
\nTjELMAkGA1UECBMCS0ExEjAQBgNVBAcTCUJhbmdhbG9yZTEMMAoGA1UEChMDU0RT
\nMQ0wCwYDVQQLEwRJU0RMMQwwCgYDVQQDEwNJQk0wggEiMA0GCSqGSIb3DQEBAQUA
\nA4IBDwAwggEKAoIBAQCl9WCOen28HAYDpj+DngMGvXBa1Z5IDCbrzQzjY7U9S5HB
\nX3b1QMEr1cSRjZlv6jTwe+yl3lUxT3ttXtyc3wWLxa66McJ2ABW57NNPRkIeNUeX
\nYvoJiqXuGZYhrORM5pE6hiWHNkKBCsPJU0jDpe6+Rajd5QhHKLWvqdp9QWKpS0wR
\n6sSERhrpyabmql5vYYpgll43ndZIXTKtQKL2BGA8mICrqVVKX7BZ6RsjqXAgC0lX
\nKiPH3eVGX8p3F8WaH+yuJTH/9rr4jQ4GWLoSYRUGwd4bYdxxLV2Inj55R61e9+Rg
\nyM7qRG9itpJUpfTbZrgmqSLyJgM1/he9HM7PlhstAgMBAAEwDQYJKoZIh*vcNAQEN
\nBQADggIBAJmCnhIN/nhp2VIgqA7td3EBD8xrejF0bT5mSUgx8flFmcCKCJh6/Oyn9
\nnl1PUp3SzSu734GdTDZiUtTXax7PYZlB3STlY0sZE7yU6zal0lIoUZEzXoohIEPVU
\nW4X3j9HF3hWDwNsuqZfQDRmndaz6NG2EPDxiWgTYXPLdYaZyTQFFe6A4tbT9gSHu
\n9UD1woFwjrSAfg03zwR7wSRSwcALsVs1BK96TYufZf+E2eFg+QBGAC5YWrZi3g4Q
\n1Xqxj5W5TwujLxSJ+8zxf6P9f0T96vGICH8Yy9AIWzUa3fXLh6tc1Pw+LbuIjEWr
\nK2TS+DLTmBAo8pQ5GsR8rShKFcPYOho2mbskAKgt4n+s63Jhu5qALS4Lw7eEQ7W7
\nqGffZ2JttNHwePAAqvx33xf+Y2SWn0fbOAlwT9BQ6ySn/qZR3e3Xl0rVqqukgCqO
\nBnQhI5WN4HkONkyaquJruTLHUlWX5T01q/yLnrRt8TCBA4qnX7HMlEmQkXiF5Poj
\nBcyCTctYu1HlijHjsWO9kztUfljI5OkVyS1q1FqcZQiziHHRiAEWbnrYn6Fgq13g
\nIws7Lw9Utogj54tPCwJ8gEkoW4eTO4tnZmPTTdWlmVhTdEjVRxE8fotztHJuVisP
\nmFCxBPWJZ8IP9t2C/4Zi1PuqXI/8YZx8LPIcQUcRxeLURIgQrpb7
\n-----END CERTIFICATE-----\n"
```

4. Remove the '\n' character from the certificate (from BEGIN to END CERTIFICATE) and store the certificate in a file.

5. Import the certificate into the keystore that was created earlier:

```
keytool -importcert -trustcacerts -alias mydomannew -file <new-certificate>
-keystore mydomain2.jks -storepass <keystore-password>
```

After rotating the client key, use the new certificate and private key to create locked vaults. On Transparent cloud tiering, update the cloud account by using the `mmcloudgateway account update` command.

Rotating Client key or revoking old certificate by using the automation script

a. Run `mcstore_lockedvaultrotateclientKey.sh` <keystorenewaliasname> <keystoreoldaliasname> <keyStorePath> <COSManagerIP> <expirationDays> <COSCACertFile>, where the first 4 parameters are mandatory and the last two parameters (<expirationDays> and <COSCACertFile>) are optional.

If the expiration date (expirationDays) is not specified, then the command will take the default expiration time, which is 365 days.

If the IBM Cloud Object Storage CA certificate (COSCACertFile) is not specified, then the CA file will be downloaded from the IBM Cloud Object Storage Manager.

b. For the description of the parameters, see the `mmcloudgateway` command.

For example, run this command:

```
./mcstore_lockedvaultrotateclientkey.sh testnew5 test /root/svt/test.ssl/test.jks 9.10.0.10
```

The system displays output similar to this:

```
Enter KeyStore Password:
Note: Before rotating the client key and certificate take a backup of old Key Store
Validating the inputs and the configuration....
COS Manager is reachable. Proceeding with Configuration...
```

```
       Transparent Cloud Tiering Server RPM already installed. Proceeding with Configuration...

       Python libraries are already installed. Proceeding with Configuration...

       CURL already installed. Proceeding with Configuration...
       Certificate stored in file </root/svt/test.ssl/test_new.crt>
       MAC verified OK
       writing RSA key
       What is your first and last name?
         [Unknown]:  demo
       What is the name of your organizational unit?
         [Unknown]:  demo
       What is the name of your organization?
         [Unknown]:  demo
       What is the name of your City or Locality?
         [Unknown]:  demo
       What is the name of your State or Province?
         [Unknown]:  demo
       What is the two-letter country code for this unit?
         [Unknown]:  IN
       Is CN=demo, OU=demo, O=demo, L=demo, ST=demo, C=IN correct? (type "yes" or "no")
         [no]:  yes

       Generating a new CSR....
       Downloading COS CA Certificate....
       Sending CSR to CleverSafe to be signed.....
         % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                        Dload  Upload   Total   Spent    Left  Speed
       100  2992  100  1777  100  1215   5758   3937 --:--:-- --:--:-- --:--:--  5769
       Retrieving Certificate from Response.....
       Importing New Client Certificate to Keystore.....
       Certificate reply was installed in keystore
       IMPORTANT: /root/svt/test.ssl contains private key, keystore and private certificate.
       You must keep a back up of /root/svt/test.ssl.
       Rotate Client Key Completed Successfully.
       Note: Please use mmcloudgateway update account command to import new certificate and private
       key in to TCT.
       New Alias Name is : testnew5
```

Updating Transparent cloud tiering with a new private key and certificate

This topic describes how to update Transparent cloud tiering with a new key and certificate.

1. Update the cloud account with the new private key and the certificate by issuing the following command:

```
   mmcloudgateway account update --cloud-nodeclass tct --acount-name mycloud
   --src-keystore-path /root/mydomain.jks --src-keystore-alias-name mydomainnew --src-keystore-type jks
   --src-keystore-pwd-file /root/pwd
```

2. For more information, see the `mmcloudgateway account update` command.

For example, to update the cloud account (node class tct, cloud name mycloud) with new key and certificate, issue the following command:

```
mmcloudgateway account update --cloud-nodeclass tct --cloud-name mycloud --src-keystore-path
/root/demold/worm\*.ssl/xyz%n\*.jks --src-keystore-alias-name wormnew --src-keystore-type jks
--src-keystore-pwd-file /root/pwd
```

The system displays output similar to this:

```
mmcloudgateway: Sending the command to the first successful node starting with c350f3u30
mmcloudgateway: This may take a while...
```

```
Note: Ensure that you have a backup of the Source Key Store used to import the private key and
certificates. Transparent Cloud Tiering removes the private key and certificate from the trust
```

```
store if the account delete command is run.
mmcloudgateway: Command completed successfully on c350f3u30.
mmcloudgateway: You can now delete the password file '/root/pwd'
mmcloudgateway: Command completed.
```

Chapter 7. Configuring file audit logging

The following topics describe various ways to configure file audit logging in IBM Spectrum Scale.

Enabling file audit logging on a file system

This topic describes how to enable file audit logging on a file system in IBM Spectrum Scale.

1. Before enabling file audit logging, the message queue must be enabled. It can be enabled automatically with the installation toolkit if the cluster contains at least three protocol nodes, or it can be manually enabled using the **mmmsgqueue** command. If you do not have three protocol nodes installed on your cluster, or want to define which nodes that the message queue servers/brokers run on, you must define the nodes to run these servers. Enable the message queue directly with the broker node list.

   Note: If the kafkaBrokerServers node class already exists, then the **-N** specified list will not be used.

   Issue a command similar to the following example:

   ```
   mmmsgqueue enable { -N NodeName[,NodeName...] | NodeFile | NodeClass
   ```

   For more information, see the *mmmsgqueue command* in *IBM Spectrum Scale: Command and Programming Reference*.

2. To enable a file system for file audit logging, issue the **mmaudit** command. If the message queue has not been previously enabled, the first invocation of **mmaudit** will also enable the message queue for the entire cluster.

   ```
   mmaudit Device enable
   ```

   For more information, see the *mmaudit command* in *IBM Spectrum Scale: Command and Programming Reference*.

   Note: If "object" is enabled on the file system that is holding the file audit log fileset, ensure that you have additional inodes defined for the file audit log fileset prior to enabling.

Disabling file audit logging on a file system

This topic describes how to disable file audit logging on a file system in IBM Spectrum Scale.

To disable file audit logging on a file system, issue the **mmaudit** command.

```
mmaudit Device disable
```

Note: The audit record fileset is not deleted during disablement.
For more information, see the *mmaudit command* in *IBM Spectrum Scale: Command and Programming Reference*.

Disabling the message queue for the cluster

This topic describes how to disable the message queue for the cluster in IBM Spectrum Scale.

To disable the message queue, you must also disable all of the file systems. Issue the **mmmsgqueue** command.

```
mmmsgqueue disable
```

For more information, see the *mmmsgqueue command* in *IBM Spectrum Scale: Command and Programming Reference*.

Actions taken when enabling the message queue and file audit logging

This topic explains the steps that occur when the user enables the message queue and file audit logging.

The message queue must be enabled and functioning before file audit logging can be enabled on a file system. If you use the installation toolkit to enable file audit logging, the toolkit will attempt to enable the message queue using protocol nodes as the primary message queue nodes. However, if protocol nodes are not available or if you want to use a different set of nodes as the primary message queue nodes, then you can use the **mmmsgqueue** command with the **-N** option and either specify a list of nodes, the path to a file containing a list of nodes, or a node class.

Enabling the message queue

The following list contains the steps that occur when you enable the message queue using the **mmmsgqueue** command with the **-N** option:

Note: If any of the following steps fail, the message queue will not be enabled.

1. Sets up the **kafkaZookeeperServers** node class using the available Linux quorum nodes that are found locally within the cluster.

   Note: If there are not at least three of these nodes with the special message queue RPMs installed on them that meet the minimum requirements given in <link_to_Linux_quorum_requirement>, this step will fail and the message queue will not be enabled.

2. Sets up the **kafkaBrokerServers** node class using nodes that are specified with the **-N** parameter.

   Note: If there are not at least three nodes in the resulting list, this step will fail and the message queue will not be enabled.

3. Modifies the default message queue server (broker) properties file and uploads the modified properties file to the CCR. Options within the properties file that are changed from the default are the list of zookeeper nodes, the port to use for listening for connections, and options for maximum size of the local disk space to use and where to store the local queue data.

4. Modifies the default zookeeper properties file and uploads the modified properties file to the CCR. Properties that are modified include the port for clients to use, the maximum number of client connections, and the default zookeeper data directory (among other options).

5. Sets up the random passwords for the brokers, consumers, and producers, and uploads the authentication configuration to the CCR.

6. Obtains the list of zookeeper nodes. For each zookeeper node, it enables the zookeeper on that node. Enabling the zookeeper node involves the following actions that take place directly on the node:

   a. Downloads the template zookeeper properties file from the CCR and places it in a local directory where it will be read whenever the zookeeper is started.

   b. Creates the unique "myid" file for this particular zookeeper in a local directory.

   c. Starts the Kafka zookeeper process by referencing the local zookeeper properties file.

7. Obtains the list of message queue (broker) nodes. For each message queue node, it enables the broker on that node. Enabling the broker involves the following actions that take place directly on the node:

   a. Downloads the template broker properties file from the CCR and places it in a local directory where it will be read whenever the broker is started.

       b. Calculates the unique broker ID and updates the local broker properties file.

       c. Reads the authentication configuration from the CCR and creates the appropriate local authentication file for the broker.

       d. Starts the Kafka broker process by referencing the local broker properties file.

8. Creates the following four IBM Spectrum Scale callbacks:

       a. Starts the Kafka broker when GPFS starts up.

       b. Starts the Kafka zookeeper when GPFS starts up.

       c. Stops the Kafka zookeeper when GPFS shuts down.

       d. Stops the Kafka broker when GPFS quorum is lost.

9. Pushes the producer authentication configuration to all eligible nodes in the cluster (nodes that are capable of adding file access events to the message queue). If less than the required minimum number of "pushes" is successful, the operation fails.

10. Updates a specific CCR variable to indicate that the message queue has been successfully configured and enabled.

Once successfully configured and enabled, the status of the message queue can be verified by running the **mmmsgqueue** command with the **status** option. This table is an example of what you would see when checking the state of the message queue:

```
# mmmsgqueue status
Node                                Contains  Broker    Contains   Zookeeper
Name                                Broker    Status    Zookeeper  Status
zipple-admin1.tuc.stglabs.ibm.com   yes       good      yes        good
zipple-admin2.tuc.stglabs.ibm.com   yes       good      yes        good
zipple-admin3.tuc.stglabs.ibm.com   yes       good      yes        good
zipple-admin4.tuc.stglabs.ibm.com   yes       good      yes        good
zipple-admin5.tuc.stglabs.ibm.com   no                  yes        good
zipple-admin6.tuc.stglabs.ibm.com   no                  yes        good
```

Enabling file audit logging

Once the message queue is enabled and started, file audit logging can be enabled for a file system.

Note:

- Like with the message queue, if any of the following steps fail, then the entire enablement of file audit logging for the given file system fails.

- However, unlike the message queue, any updates to the configuration or filesets will be rolled back so that the cluster is not left in a state where a file system is partially enabled for file audit logging.

1. Verifies that the message queue is successfully configured and enabled.

2. If the consumer node class does not exist, it creates it based off of the Kafka broker node class. In addition, the consumer authentication configuration information is pushed to all of the nodes in the newly created consumer node class.

3. For each message queue server (broker) node in the cluster, verifies that the minimum amount of required local disk space is available to enable file audit logging for a file system device. Depending on the number of message queue server (broker) nodes in the cluster, this check might take some time because each node has to be queried.

4. For each consumer node in the consumer node class, verifies that the sink file system is mounted on the node. The sink file system is the file system where the file audit logging fileset is located that contains the audit log files.

5. Updates the audit configuration with the file audit logging configuration information including the file system that will be audited, the device of the sink fileset, and the sink fileset name among other attributes.

6. Adds the topic to the message queue representing the file system that will be audited.

7. Creates the access control rules so that the producers and consumers can access the newly created topic in the message queue that represents the file system that is being audited. This allows the producers to write events to the queue and consumers to read events from the queue for the specific topic.

8. If needed, it creates the sink fileset in IAM noncompliant mode on the device that is specified in the configuration.

9. Enables the consumer group, which consists of the following actions on each of the consumer nodes:

   a. Generates the system service file that is specific to the consumer process for the given topic and file system that will be audited.

   b. Reloads the system daemon to allow the new system service file to be discovered.

   c. Creates a lock file that must be removed by the consumer process to show that the consumer process has started completely and successfully.

   d. Starts the service specified by the system service file.

   e. Registers the sink file system used by the consumer process for later preumount call back use.

10. Creates the policy partitions used to receive LWE events and block LWE events from certain filesets and paths:

    a. Creates the policy partition used to receive LWE events for the file system device being audited.

    b. If needed, creates policy partition(s) to skip file system operations in the file audit logging sink fileset.

    c. If needed, creates global policy partition(s) to skip known paths (such as the CES shared root) where LWE events are not wanted.

11. If this is the first file system that will be audited, creates the file audit logging IBM Spectrum Scale callbacks for the consumer processes:

    a. Creates a IBM Spectrum Scale callback to start the consumer process on the consumer node if the sink file system is mounted.

12. Makes the change to the file system configuration so that the `mmlsfs` command with the `--file-audit-log` option shows as yes.

To verify the settings of one or more file systems that are enabled for file audit logging, run the following command that will list configuration information for all file systems configured for file audit logging:

```
mmaudit all list
```

This is an example with output:

```
# mmaudit all list
Audit     Cluster                                  Fileset   Fileset             Retention
Device    ID                                       Device    Name                (Days)
--------------------------------------------------------------------------------------------
fs0       11430652110844333597                     fs1       john1               25
fs1       11430652110844333597                     fs1       john2               25
fs2       11430652110844333597                     fs1       john1               25
```

Chapter 8. Configuring Active File Management

The following topics list the parameters required to configure Active File Management.

Configuration parameters for AFM

The following table lists the AFM and AFM-based DR configuration parameters with their default values and the commands to set and edit the parameters.

Table 7. Configuration parameters at cache and their default values at the cache cluster

AFM configuration parameter	Unit	Description	Mode on AFM
`afmAsyncDelay`	seconds	Indicates the time at which the requests start flushing to the home cluster. For write-intensive applications that write to the same set of files, this delay is helpful because it replaces multiple writes to the home cluster with a single write containing the latest data. However, setting a very high value weakens the consistency of the data on the remote cluster.	SW, IW
`afmDirLookupRefreshInterval`	seconds	Defines the frequency of revalidation triggered by a look-up operation such as `ls` or `stat` on a directory from the cache cluster. AFM sends a message to the home cluster to find out whether the metadata of that directory is modified since it was last revalidated. If so, the latest metadata information at the home cluster is reflected on the cache cluster.	RO, LU, IW
`afmDirOpenRefreshInterval`	seconds	Defines the frequency of revalidations triggered by the read and update operations on a directory from the cache cluster. AFM sends a message to the home cluster to find whether the metadata of that directory is modified since it was last revalidated. Open requests on files or sub-directories on that directory are served from the cache fileset until the `afmDirOpenRefreshInterval` expires, after which the open requests are sent to the home cluster.	RO, LU, IW
`afmDisconnectTimeout`	seconds	Defines the interval until which the MDS waits after it detects that the home cluster is inaccessible before declaring the outage by moving the cache cluster state to Disconnected.	RO, SW, IW, LU

Table 7. Configuration parameters at cache and their default values at the cache cluster (continued)

AFM configuration parameter	Unit	Description	Mode on AFM
`afmExpirationTimeout`	seconds	Is used with `afmDisconnectTimeout` to control the duration of a network outage between the cache and home clusters before the data in the cache expires and becomes unavailable until a home reconnection occurs.	RO
`afmFileLookupRefreshInterval`	seconds	Defines the frequency of revalidation triggered by a look-up operation on a file such as ls or stat, from the cache cluster. AFM sends a message to the home cluster to determine that the metadata of the file is modified since it was last revalidated. If so, the latest metadata information at home is reflected on the cache cluster.	RO, LU, IW
`afmFileOpenRefreshInterval`	seconds	Defines the frequency of revalidations triggered by the read and write operations on a file from the cache cluster. AFM sends a message to the home cluster to determine that the metadata of the file is modified since it was last revalidated. Open requests on the file are served from the cache fileset until the `afmFileOpenRefreshInterval` expires, after which the open requests are sent to the home cluster.	RO, LU, IW
`afmEnableAutoEviction`	-	Indicates if automatic eviction is triggered on a fileset.	RO, SW, IW,LU
`afmPrefetchThreshold`	Whole number - 0 to 100	Controls the partial file caching feature. 0: Full file prefetching after reading 3 blocks 1-99: The percentage of the file size that must be cached before the entire file is pulled into the cache cluster 100: Disables full file prefetching. Only fetches and caches data blocks that are read by the application. When all data blocks have been cached, the file is marked as cached.	RO, SW, IW,LU
`afmShowHomeSnapshot`	-	Controls the visibility of the home snapshot directory in the cache cluster. For this to be visible in the cache cluster, this variable must to be set to yes, and the snapshot directory name in the cache and home clusters must not be the same.	RO, LU

AFM configuration parameter	Unit	Description	Mode on AFM
`afmReadSparseThreshold`	Bytes	When a sparse file at the home cluster is read into the cache, the cache cluster maintains the sparseness, if the size of the file exceeds the `afmReadSparseThreshold`. If the size of a file is less than the threshold, sparseness is not maintained at the cache cluster.	RO, SW, IW,LU
`afmHashVersion`		Specifies the older version 2 or newer version 4 of the gateway node hashing algorithm, such as, the `mmchconfig afmHashVersion=2` or `mmchconfig afmHashVersion=4`, to minimize the impact of gateway nodes joining or leaving the active cluster by running as few recoveries as possible and balance the mapping of AFM filesets across gateway node. Default `afmHashVersion` is 2.	Not applicable
`afmMaxParallelRecoveries`	Whole number	Specifies the number of filesets in the cluster on all filesystems, on which recovery is run.	RO, SW, IW,LU
`afmMountRetryInterval`	seconds	Specifies the interval after which the primary gateway retries an operation at home, in cases where home is in an unhealthy state (see Unmounted, Dropped states). Needs the GW node recycle or `-i` option for immediate effect.	RO, SW, IW,LU
`afmAsyncOpWaitTimeout`	seconds	Specifies the time that AFM waits for completion of any inflight asynchronous operation which is synchronizing with the home cluster. Subsequently, AFM cancels the operation and synchronizes again after home is available.	RO, SW, IW
`afmSyncOpWaitTimeout`	seconds	Specifies the time that AFM waits for completion of any inflight synchronous operation which is synchronizing with the home cluster. When any application is performing any synchronous operation at cache, AFM tries to get a response from home. If home is not responding, application might be unresponsive. If operation does not complete in this timeout interval, AFM cancels the operation.	RO, SW, IW

Table 7. Configuration parameters at cache and their default values at the cache cluster (continued)

AFM configuration parameter	Unit	Description	Mode on AFM
`afmRevalOpWaitTimeout`	seconds	Specifies the time that AFM waits for completion of revalidation to get response from the home cluster. Revalidation checks if any changes are available at home (data and metadata) that need to be updated to the cache cluster. Revalidation is performed when application trigger operations like lookup or open at cache. If revalidation is not completed within this time, AFM cancels the operation and returns data available at cache to the application.	RO, IW

Table 8. Configuration parameters at cache and their default values at the cache cluster - Valid values

AFM configuration parameter	Valid values	Default value	Tunable at the cluster level	Tunable at the fileset level
`afmAsyncDelay`	1 - 2147483647	15	Yes	Yes
`afmDirLookupRefreshInterval`	0 - 2147483647	60	Yes	Yes
`afmDirOpenRefreshInterval`	0 - 2147483647	60	Yes	Yes
`afmDisconnectTimeout`	0 - 2147483647, disable	60	Yes	No
`afmExpirationTimeout`	0 - 2147483647, disable	disabled	Yes	Yes
`afmFileLookupRefreshInterval`	0 - 2147483647	30	Yes	Yes
`afmFileOpenRefreshInterval`	0 - 2147483647	30	Yes	Yes
`afmEnableAutoEviction`	Yes | No	Yes	No	Yes
`afmPrefetchThreshold`	0 - 100	0	No	Yes
`afmShowHomeSnapshot`	Yes | No	No	Yes	Yes
`afmReadSparseThreshold`	0 - 2147483647	128 M	Yes	Yes
`afmHashVersion`	1 | 2 | 4	2	Yes	No
`afmMaxParallelRecoveries`	0 - 128	0	Yes	No
`afmMountRetryInterval`	1 - 2147483647	300	No	No
`afmAsyncOpWaitTimeout`	5 - 2147483647	300	Yes	No
`afmSyncOpWaitTimeout`	5 - 2147483647	180	Yes	No
`afmRevalOpWaitTimeout`	5 - 2147483647	180	Yes	No

Parallel I/O configuration parameters for AFM

The parameters in the following table can be used at the cache cluster for tuning parallel I/O. All parallel I/O parameters do not need the gateway node daemon recycle for the new value to take effect.

Table 9. Configuration parameters at cache for parallel I/O

AFM configuration parameter	Unit	Description	Mode on AFM
`afmNumFlushThreads`	Whole number	Defines the number of threads used on each gateway to synchronize updates to the home cluster. The default value is 4, which is sufficient for most installations. The current maximum value is 1024, which is too high for most installations. Ensure that you do not set this parameter to a very high value.	SW, IW
`afmNumReadThreads`	Whole number	Defines the number of threads used on each participating gateway node during a parallel read. The default value of this parameter is 1. That is, one reader thread is active on every gateway node for each big read operation qualifying for splitting as per the parallel read threshold value.	SW, IW,RO, LU
`afmNumWriteThreads`	Whole number	Defines the number of threads used on each participating gateway node during a parallel write. The default value of this parameter is 1. That is, one writer thread is active on every gateway node for each big write operation qualifying for splitting as per the parallel write threshold value.	SW, IW
`afmParallelReadChunkSize`	MB	Defines the minimum chunk size of the read that needs to be distributed among the gateway nodes during parallel reads. Values are in bytes.	SW, IW, RO, LU
`afmParallelReadThreshold`	MB	Defines the threshold beyond which parallel reads become effective. Reads are split into chunks when the file size exceeds this threshold value. Values are in MB. The default value is 1024 MB.	SW, IW, RO, LU
`afmParallelWriteChunkSize`	MB	Defines the minimum chunk size of the write that needs to be distributed among the gateway nodes during parallel writes. Values are in bytes.	SW, IW
`afmParallelWriteThreshold`	MB	Defines the threshold beyond which parallel writes become effective. Writes are split into chunks when file size exceeds this threshold value. Values are in MB. The default value is 1024 MB.	SW, IW

Table 9. Configuration parameters at cache for parallel I/O (continued)

AFM configuration parameter	Unit	Description	Mode on AFM
`afmHardMemThreshold`	Byte	Sets the maximum memory that AFM can use on each gateway node for handling queues. After this limit is reached, queues might not be handled on this gateway node due to lack of sufficient memory. Filesets belonging to this gateway node might go to a 'Dropped' state, depending on the activity. Exceeding the limit can occur if the cache cluster is disconnected for an extended time or if the connection with the home cluster has low bandwidth and a lot pending requests are accumulated in the queue. After the value of this parameter is changed, a gateway node daemon recycle is required for the new value to take effect.	

Table 10. Configuration parameters at cache for parallel I/O - valid values

AFM configuration parameter	Valid values	Default value	Tunable at the cluster level	Tunable at the fileset level
`afmNumFlushThreads`	1 - 1024	4	No	Yes
`afmNumReadThreads`	1 - 64	1	Yes	Yes
`afmNumWriteThreads`	1 - 64	1	Yes	Yes
`afmParallelReadChunkSize`	0 - 2147483647	128	Yes	Yes
`afmParallelReadThreshold`	0 - 2147483647	1024	Yes	Yes
`afmParallelWriteChunkSize`	0 - 2147483647	128	Yes	Yes
`afmParallelWriteThreshold`	0 - 2147483647	1024	Yes	Yes
`afmHardMemThreshold`		5 G	Yes	No

Chapter 9. Configuring AFM-based DR

The following topics list the parameters required to configure AFM-based DR.

Configuration parameters for AFM-based DR

The following table lists the AFM-based DR configuration parameters with their default values and the commands to set and edit the parameters:

Table 11. Configuration parameters at primary and their default values

AFM configuration parameter	Unit	Description	Mode on AFM-DR
afmAsyncDelay	Seconds	Indicates the time at which the requests start flushing to the home cluster. For write-intensive applications that write to the same set of files, this delay is helpful because it replaces multiple writes to the home cluster with a single write containing the latest data. However, setting a very high value weakens the consistency of the data on the remote cluster.	primary
afmDisconnectTimeout	Seconds	Defines the interval until which the MDS waits after it detects that the home cluster is inaccessible before declaring the outage by moving the cache cluster state to Disconnected.	primary
mountRetryInterval	Seconds	Specifies the interval after which the MDS retries an operation at home, in cases where the home is in an unhealthy state. Needs the GW node recycle or `-i` option for immediate effect. It is tunable per gateway. The updated value is visible only as long as the gateway is active and is applicable for all filesets owned by that gateway.	primary
afmRPO	Minutes, hours, weeks	Specifies the interval for generating RPO snapshots.	primary
afmAsyncOpWaitTimeout	seconds	Specifies the time that AFM DR waits for completion of any inflight asynchronous operation which is synchronizing with the primary. Subsequently, AFM DR cancels the operation and synchronizes again after primary is available.	primary

Table 11. Configuration parameters at primary and their default values (continued)

AFM configuration parameter	Unit	Description	Mode on AFM-DR
afmSyncOpWaitTimeout	seconds	Specifies the time that AFM DR waits for completion of any inflight synchronous operation which is synchronizing with the primary. When any application is performing any synchronous operation at cache, AFM DR tries to get a response from primary. If primary is not responding, application might be unresponsive. If operation does not complete in this timeout interval, AFM DR cancels the operation.	primary

Table 12. Configuration parameters at primary and their default values - Valid values

AFM configuration parameter	Valid values	Default value	Tunable at the cluster level	Tunable at the fileset level
afmAsyncDelay	1 - 2147483647	15	Yes	Yes
afmDisconnectTimeout	0 - 2147483647, disable	60	Yes	No
mountRetryInterval		300	No	No
afmRPO	720 - 2147483647	disabled	No	Yes
afmAsyncOpWaitTimeout	5 - 2147483647	300	Yes	No
afmSyncOpWaitTimeout	5 - 2147483647	180	Yes	No

Parallel I/O configuration parameters for AFM-based DR

All parallel I/O parameters do not need the gateway node daemon recycle for the new value to take effect.

The parameters in the following table can be used at the cache cluster for tuning parallel I/O:

Table 13. Configuration parameters at cache for parallel I/O

AFM configuration parameter	Unit	Description	Mode on AFM DR
afmNumFlushThreads	Whole number	Defines the number of threads used on each gateway to synchronize updates to the home cluster. The default value is 4, which is sufficient for most installations. The current maximum value is 1024, which is too high for most installations. Do not set this parameter to an extreme value.	primary
afmNumWriteThreads	Whole number	Defines the number of threads used on each participating gateway node during a parallel write. The default value of this parameter is 1. That is, one writer thread is active on every gateway node for each big write operation qualifying for splitting as per the parallel write threshold value.	primary

Table 13. Configuration parameters at cache for parallel I/O (continued)

AFM configuration parameter	Unit	Description	Mode on AFM DR
`afmParallelWriteChunkSize`	MB	Defines the minimum chunk size of the write that needs to be distributed among the gateway nodes during parallel writes. Values are in bytes.	primary
`afmParallelWriteThreshold`	MB	Defines the threshold beyond which parallel writes become effective. Writes are split into chunks when file size exceeds this threshold value. Values are in MB. The default value is 1024 MB.	primary
`afmHardMemThreshold`	Byte	Sets the maximum memory that AFM can use on each gateway node for handling queues. After this limit is reached, queues might not be handled on this gateway node due to lack of sufficient memory. Filesets belonging to this gateway node might go to a 'Dropped' state, depending on the activity. Exceeding the limit can occur if the cache cluster is disconnected for an extended time or if the connection with the home cluster has low bandwidth and therefore a lot of pending requests are accumulated in the queue. After the value of this parameter is changed, a gateway node daemon recycle is required for the new value to take effect.	

Table 14. Configuration parameters at cache for parallel I/O - valid values

AFM configuration parameter	Valid values	Default value	Tunable at the cluster level	Tunable at the fileset level
`afmNumFlushThreads`	1 - 1024	4	No	Yes
`afmNumWriteThreads`	1 - 64	1	Yes	Yes
`afmParallelWriteChunkSize`	0 - 2147483647	128	Yes	Yes
`afmParallelWriteThreshold`	0 - 2147483647	1024	Yes	Yes
`afmHardMemThreshold`		5 G	Yes	No

Chapter 10. Tuning for Kernel NFS backend on AFM and AFM DR

If AFM communication use the NFSv3 protocol, for peak performance, tune the gateway NFS servers that host home exports and the gateway servers that support cache/primary filesets.

Most of these tuning parameters require at least the AFM client to be restarted. Ensure that the NFS server is not mounted. Unlink the AFM fileset, or stop GPFS or IBM Spectrum Scale on the gateway node.

Tuning for 1GigE networks is different from tuning 10GigE networks. For 10GigE, all settings need to be scaled up, but not necessarily by a factor of 10. Many of these settings are affected after a server reboot. Therefore, each time a server restarts, the settings must be reset. The TCP buffer tuning is required for all 10GigE links and for 1GigE links where the value of `RTT` is greater than 0.

Tuning the gateway node on the NFS client

To tune the gateway node on the NFS client, you can set the maximum number of (TCP) RPC requests that can be in flight by using the `sunrpc.tcp_slot_table_entries` or `/proc/sys/sunrpc/tcp_slot_table_entries` parameter.

For 1 GigE, if there is no round-trip time, leave the default value. You can increase the value to a number greater than 16 if the round-trip time is large. For 10 GigE, ensure that this value is 48 or a number greater than 48 depending on the round-trip time.

When you set the `seqDiscardThreshold` parameter, it affects AFM or AFM DR as follows:
- If I/O requests are from a node that is not the gateway node, there is no effect.
- If the read request is on the gateway node for an uncached file, a higher `seqDiscardThreshold` value results in better performance as it allows the gateway to cache more data. When the data is returned to the application, there is a greater chance that it comes out of the cache/primary cluster.

Tuning on both the NFS client (gateway) and the NFS server (the home/secondary cluster)

This topic describes the tuning on both the NFS client (gateway) and the NFS server (the home/secondary cluster).

You must set TCP values that are appropriate for the delay (`buffer size = bandwidth * RTT`).

For example, if your ping time is 50 ms, and the end-to-end network consists of all 100BT Ethernet and OC3 (155 Mbps), the TCP buffers must be the following: `0.05 sec * 10 MB/sec = 500 KB`

If you are connected using a T1 line (1 Mbps) or less, do not change the default buffers. Faster networks usually benefit from buffer tuning.

The following parameters can also be used for tuning. A 12194304 buffer size is provided here as an example value for a 1 GigE link with a delay of 120ms. To set these values, set the following configurations in a file and load it with `sysctl -p` filename.

The following are example values. Initial testing is required to determine the best value for a particular system:

```
net.ipv4.tcp_rmem = 12194304 12194304 12194304
        net.ipv4.tcp_wmem = 12194304 12194304 12194304
        net.ipv4.tcp_mem = 16777216 16777216 16777216
        net.core.rmem_max = 12194304
        net.core.wmem_max = 12194304
        net.core.rmem_default = 12194304
        net.core.wmem_default = 12194304
        net.core.optmem_max = 12194304
        net.core.netdev_max_backlog = 250000
        net.ipv4.tcp_no_metrics_save = 1
        net.ipv4.tcp_timestamps = 0
        net.ipv4.tcp_sack = 1
```

Note: For TCP tuning, the **sysctl** value changes do not take effect until a new TCP connection is created, which occurs at NFS mount time. Therefore, for TCP changes, it is critical that the AFM fileset and the NFS client are unmounted and GPFS is shut down.

With Red Hat Enterprise Linux 6.1 and later, both the NFS client and the server perform TCP auto-tuning. It automatically increases the size of the TCP buffer within the specified limits through **sysctl**. If the client or the server TCP limits are too low, the TCP buffer grows for various round-trip time between the GPFS clusters. With Red Hat Enterprise Linux 6.1 and earlier, NFS is limited in its ability to tune the TCP connection. Therefore, do not use a version earlier than Red Hat Enterprise Linux 6.1 in the cache/primary cluster.

As a GPFS cluster might be handling local and remote NFS clients, you can set the GPFS server values for the largest expected round-trip time of any NFS client. This ensures that the GPFS server can handle clients at various locations. Then on the NFS clients, set the TCP buffer values that are appropriate for the SONAS cluster that they are accessing.

The gateway node is both an NFS server for standard NFS clients if they exist and an NFS client for communication with the home/secondary cluster. Ensure that the TCP values are set appropriately, because values that are either too high or too low can negatively impact performance.

If performance continues to be an issue, increase the buffer value by up to 50%. If you increase the buffer value by more than 50%, it might have a negative effect.

Tuning the NFS server on the home/secondary cluster or the NFS server

This topic describes the tuning on the NFS server on the home/secondary cluster or the NFS server.

Table 15. NFS server parameters

Parameter name with path	Description
/proc/fs/nfsd/max_block_size	Set to 1 MB for improved performance.
/proc/fs/nfsd/threads	Set to a minimum value of 32. You can set this value to greater than 128 depending on the throughput capacity and the round-trip time between the cache/primary and home/secondary clusters. Determining the correct value might take a few trials.

Table 15. NFS server parameters (continued)

Parameter name with path	Description
nfsPrefetchStrategy	Set it to a number between 5 and 10. As AFM uses NFS, ensure that this is set on the home/secondary GPFS cluster. After the NFS values are set, you can mount and access the AFM filesets. The first time the fileset is accessed the AFM NFS client mounts the home/secondary server or servers. To see these mounts on a gateway node, enter the following command: `cat /proc/mounts`. The system displays the mount point and the mount options. If the `wsize` and `rsize` values are not 1 MB, you can adjust the parameters and mount the AFM filesets again to get the correct values.

Chapter 11. Performing GPFS administration tasks

Before you perform GPFS administration tasks, review topics such as getting started with GPFS, requirements for administering a GPFS file system, and common command principles.

For information on getting started with GPFS, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*. This includes:
1. Installing GPFS
2. GPFS cluster creation considerations
3. Configuring and tuning your system for GPFS
4. Starting GPFS
5. Network Shared Disk creation considerations
6. File system creation considerations

The information for administration and maintenance of GPFS and your file systems is covered in topics including:
1. "Requirements for administering a GPFS file system" and "Common GPFS command principles" on page 103
2. Chapter 1, "Configuring the GPFS cluster," on page 1
3. Chapter 13, "Managing file systems," on page 109
4. Chapter 15, "Managing disks," on page 157
5. Chapter 20, "Managing GPFS quotas," on page 285
6. Chapter 22, "Managing GPFS access control lists," on page 303
7. **Command reference** in *IBM Spectrum Scale: Command and Programming Reference*
8. **GPFS programming interfaces** in *IBM Spectrum Scale: Command and Programming Reference*
9. **GPFS user exits** in *IBM Spectrum Scale: Command and Programming Reference*
10. Chapter 23, "Considerations for GPFS applications," on page 335
11. Chapter 14, "File system format changes between versions of IBM Spectrum Scale," on page 153

Requirements for administering a GPFS file system

Root authority is required to perform all GPFS administration tasks except those with a function limited to listing certain GPFS operating characteristics or modifying individual user file attributes.

On Windows, root authority normally means users in the Administrators group. However, for clusters with both Windows and UNIX nodes, only the special Active Directory domain user **root** qualifies as having root authority for the purposes of administering GPFS. For more information on GPFS prerequisites, see the topic *Installing GPFS prerequisites* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The GPFS commands are designed to maintain the appropriate environment across all nodes in the cluster. To achieve this goal, the GPFS commands use the remote shell and remote file copy commands that you specify on either the **mmcrcluster** or the **mmchcluster** command.

The default remote commands are **ssh** and **scp**, but you can designate any other remote commands provided they have compatible syntax.

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

You can avoid configuring your GPFS nodes to allow remote access to the root user ID, by using sudo wrapper scripts to run GPFS administrative commands. See "Running IBM Spectrum Scale commands without remote root login" on page 17.

GPFS does not need to know which nodes are being used for administration purposes. It is the administrator's responsibility to issue **mm** commands only from nodes that are properly configured and can access the rest of the nodes in the cluster.

Note: If your cluster includes Windows nodes, you must designate **ssh** and **scp** as the remote communication program.

adminMode configuration attribute

GPFS recognizes the **adminMode** configuration attribute. It specifies whether all nodes in the cluster will be used for issuing GPFS administration commands or just a subset of the nodes.

The **adminMode** attribute is set with the **mmchconfig** command and can have one of two values:

`allToAll`
    Indicates that all nodes in the cluster can be used for running GPFS administration commands and that all nodes are able to execute remote commands on any other node in the cluster without the need of a password.

    The major advantage of this mode of operation is that GPFS can automatically recover missing or corrupted configuration files in almost all circumstances. The major disadvantage is that all nodes in the cluster must have root level access to all other nodes.

`central`
    Indicates that only a subset of the nodes will be used for running GPFS commands and that only those nodes will be able to execute remote commands on the rest of the nodes in the cluster without the need of a password.

    The major advantage of this mode of administration is that the number of nodes that must have root level access to the rest of the nodes is limited and can be as low as one. The disadvantage is that GPFS may not be able to automatically recover from loss of certain configuration files. For example, if the SSL key files are not present on some of the nodes, the operator may have to intervene to recover the missing data. Similarly, it may be necessary to shut down GPFS when adding new quorum nodes. If an operator intervention is needed, you will see appropriate messages in the GPFS log or on the screen.

    Note List:
    1. Any node used for the IBM Spectrum Scale GUI is considered as an administrative node and must have the ability to execute remote commands on all other nodes in the cluster without the need of a password as the root user or as the configured gpfs admin user.

2. If the GPFS cluster is configured to support Clustered NFS (CNFS), all CNFS member nodes must belong to the subset of nodes that are able to execute remote commands without the need of a password as the root user or as the configured gpfs admin user.

3. If the GPFS cluster is configured to support Clustered export services (CES), all CES member nodes must belong to the subset of nodes that are able to execute remote commands without the need of a password as the root user or as the configured gpfs admin user.

4. The IBM Spectrum Scale REST API must be configured on nodes that are able to execute remote commands without a password as the root user or as the configured gpfs admin user.

5. If a IBM Spectrum Protect server is used to back up the GPFS file system data, the nodes that are used as IBM Spectrum Protect clients must belong to the subset of nodes that are able to execute remote commands without the need of a password as the root user or as the configured gpfs admin user.

6. Windows GPFS clusters typically use **central** mode. **allToAll** mode requires that the GPFS Administrative service (mmwinserv) be configured to run as the special domain root account. For more information, see *Procedure for installing GPFS on Windows nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

7. If Call home is configured to execute daily/weekly data gathering or the autoconfig option is to be used, the call home node (also known as call home server) must be able to reach call home clients without a password for scp data transfer as the root user or as the configured gpfs admin user.

Clusters created with the GPFS 3.3 or later level of the code have **adminMode** set to **central** by default. Clusters migrated from GPFS 3.2 or earlier versions will continue to operate as before and will have **adminMode** set to **allToAll**.

You can change the mode of operations at any time with the help of the **mmchconfig** command. For example, to switch the mode of administration from **allToAll** to **central**, issue:

```
mmchconfig adminMode=central
```

Use the **mmlsconfig adminMode** command to display the mode of administration currently in effect for the cluster.

Common GPFS command principles

There are some common principles that you should keep in mind when you are running GPFS commands.

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

Specifying nodes as input to GPFS commands

Many GPFS commands accept a node or multiple nodes as part of their input, using the **-N** flag.

Nodes can be specified to GPFS 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 GPFS 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.

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 GPFS command supports all of the node specification options described in this topic. To learn what kinds of node specifications are supported by a particular GPFS command, see the relevant command description in *Command reference* in *IBM Spectrum Scale: Command and Programming Reference*.

Stanza files

The input to a number of GPFS commands can be provided in a file organized in a stanza format.

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

GPFS 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.

For more information about the IBM Spectrum Scale RAID commands that use stanzas, see *IBM Spectrum Scale RAID: Administration* in Elastic Storage Server (ESS) documentation on IBM Knowledge Center.

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
```

Note: The server name used in the NSD stanza file must be resolvable by the system.

Listing active IBM Spectrum Scale commands

You can list the active IBM Spectrum Scale commands that are running on the file system manager node.

Most IBM Spectrum Scale commands run within the GPFS daemon on the file system manager node. Even if you start a command on another node of the cluster, the node typically sends the command to the file system manager node to be executed. (Two exceptions are the **mmdiag** command and the **mmfsadm dump** command, which run on the node where they were started.)

To list the active commands on the file system manager node, follow these steps:

1. Enter the **mmlsmgr** command with no parameters to discover which node is the file system manager node. For more information on other options available for the `mmlsmgr` command, see *mmlsmgr command* in *IBM Spectrum Scale: Command and Programming Reference* guide. In the following example, the **mmlsmgr** command reports that node05 is the file system manager node:

```
# mmlsmgr
file system      manager node
---------------- ------------------
gpfs1            192.168.145.14 (node05)

Cluster manager node: 192.168.145.13 (node03)
```

2. Go to the command console on the file system manager node and enter **mmdiag --commands**:

```
# mmdiag --commands
=== mmdiag: commands ===
CrHashTable 0x1167A28F0 n 2
  cmd sock 24 cookie 2233688162 owner 38076509 id 0x3FE6046C2700000D(#13) uses 1
type 76 start 1460415325.957724 flags 0x106 SG none line 'mmdiag --commands'
  cmd sock 12 cookie 521581069 owner 57606185 id 0x3FE6046C2700000C(#12) uses 1
type 13 start 1460415323.336314 flags 0x117 SG gpfs1 line 'mmrestripefs /dev/business1 -m'
```

The output indicates that two commands are running: the **mmdiag --commands** command that you just entered and the **mmrestripefs** command, which was started from another node.

Note: The output contains two lines about active commands. Each line begins with the term `cmd` and wraps to the next line. You might be interested in the following fields:

start   The system time at which the command was received.

SG   The name of the file system, or `None`.

line   The command as received by the GPFS daemon.

The remaining input is detailed debugging data that is used for product support. For more information on `mmdiag` command output, see the topic *mmdiag command* in the *IBM Spectrum Scale: Command and Programming Reference* guide.

Chapter 12. Verifying network operation with the mmnetverify command

Verify network operation with the **mmnetverify** command.

Important: Proper operation of IBM Spectrum Scale depends on reliable TCP/IP communication among the nodes of a cluster. Before you create or reconfigure an IBM Spectrum Scale cluster, ensure that proper host name resolution and ICMP echo (network ping) are enabled among the nodes.

With the **mmnetverify** command, you can do many types of network checks either before or after you create or reconfigure a cluster. Run the command beforehand to verify that the nodes can communicate properly. Run the command afterward at any time to verify communication or to analyze a network problem. For more information, see the topic *mmnetverify command* in the *IBM Spectrum Scale: Command and Programming Reference*.

The **mmnetverify** command uses the concepts of local nodes and target nodes. A *local node* is a node from which a network test is run. You can enter the command on one node and have it run on multiple separate local nodes. A *target node* is a node against which a test is run.

You can run tests on one node against multiple nodes. The following command runs tests on node1 against node2 and then on node1 against node3:

```
mmnetverify connectivity --N node1 --target-nodes node2,node3
```

You can also run tests on multiple nodes against multiple nodes. The following command runs tests on node1 against node1 and node2 and then on node2 against node1 and node2:

```
mmnetverify connectivity --N node1,node2  --target-nodes node1,node2
```

It is not necessary to enter the command from a node that is involved in testing. For example, you can run the following command from node1, node2, node3, or any other node in the cluster:

```
mmnetverify data --N node1 --target-nodes node2,node3
```

To run tests against all the nodes in the cluster, omit the `--target-nodes` parameter (example 1). Similarly, to run the test on all the nodes in the cluster, omit the `--N` parameter (example 2):

```
(1) mmnetverify data-medium --N node1
(2) mmnetverify data-medium --target-nodes node2,node3,node4
```

To run all the tests, omit the test parameter:

```
mmnetverify --N node1 --target-nodes node2,node3,node4
```

The groups of tests include connectivity, port, data, bandwidth, and flood tests. You can run tests individually or as a group. For example, you can run resolution, ping, shell, and copy tests individually, or you can run all of them by specifying the keyword `connectivity`.

The command writes the results of tests to the console by default, or to a log file as in the following example:

```
mmnetverify port --N node1 --target-nodes all --log-file results.log
```

If you are running tests against nodes that are not organized into a cluster, you must specify the nodes in a configuration file. The file must at minimum contain a list of the nodes in the test. You must also include the node from which you are starting the command:

```
node node_starting
node node1
node node2
node node3
node node4
```

Run the command in the usual way and include the configuration file:

```
mmnetverify ping --N node1,node2,node3,node4 --target-nodes
node1,node2,node3,node4  --configuration-file config.txt
```

You can also use the configuration file for other purposes, such as specifying a nondefault shell command or file copy command.

Related information:

See the topic *mmnetverify command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Chapter 13. Managing file systems

There are several file system management tasks outlined in this topic.

For information on how to create GPFS file systems, see *A sample file system creation* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* and the *mmcrfs command*.

File system management tasks include:
1. "Mounting a file system"
2. "Unmounting a file system" on page 112
3. "Deleting a file system" on page 113
4. "Determining which nodes have a file system mounted" on page 114
5. "Checking and repairing a file system" on page 114
6. "Listing file system attributes" on page 116
7. "Modifying file system attributes" on page 117
8. "Querying and changing file replication attributes" on page 118
9. "Using Direct I/O on a file in a GPFS file system" on page 119
10. "Restriping a GPFS file system" on page 128
11. "Querying file system space" on page 129
12. "Querying and reducing file system fragmentation" on page 130
13. "Protecting data in a file system using backup" on page 132
14. "Scale Out Backup and Restore (SOBAR)" on page 139

Managing filesets, storage pools and policies is also a file system management task. For more information on managing storage pools, filesets and policies, see Chapter 25, "Information lifecycle management for IBM Spectrum Scale," on page 355. Use the following information to manage file systems in IBM Spectrum Scale.

Managing file system through GPFS GUI

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Files** > **File Systems**. For more information on managing file systems through GUI, see "Creating and managing file systems using GUI" on page 147.

Mounting a file system

You must explicitly mount a GPFS file system if this is the first time the file system is being mounted after its creation, or you specified *not to* automatically mount (**-A no**) the file system when you created it.

If you allowed the default value for the automatic mount option (**-A yes**) when you created the file system, then you do not need to use this procedure after restarting GPFS on the nodes.

To mount a GPFS file system, enter:
```
mmmount device
```

where *device* is the name of the file system. For example, to mount the file system **fs1**, enter:
```
mmmount fs1
```

Mounting a file system on multiple nodes

This topic describes how to mount a file systems on multiple nodes.

To mount file system **fs1** on all nodes in the GPFS cluster, issue this command:

```
mmmount fs1 -a
```

To mount a file system only on a specific set of nodes, use the **-N** flag of the **mmmount** command.

Mount options specific to IBM Spectrum Scale

Mount options specific to IBM Spectrum Scale can be specified with the **-o** parameter on the **mmchfs**, **mmremotefs**, **mmmount** and **mount** commands. Options specified with the **mmchfs** and **mmremotefs** commands are recorded in the GPFS configuration files and are passed as default options to subsequent mount commands on all nodes in the cluster. Options specified with the **mmmount** or **mount** commands override the existing default settings and are not persistent.

All of the mount options can be specified using the **-o** parameter. Multiple options should be separated only by a comma. If an option is specified multiple times, the last instance is the one that takes effect. Certain options can also be set with specifically designated command flags. Unless otherwise stated, mount options can be specified as:

option or *option*=1 or *option*=**yes** - to enable the option

no*option* or *option*=0 or *option*=**no** - to disable the option

The *option*={1 | 0 | **yes** | **no**} syntax should be used for options that can be intercepted by the **mount** command and not passed through to GPFS. An example is the **atime** option in the Linux environment.

The GPFS-specific mount options are:

atime
> Update inode access time for each access. This option can also be controlled with the **-S** option of the **mmcrfs** and **mmchfs** commands.

mtime
> Always return accurate file modification times. This is the default. This option can also be controlled with the **-E** option on the **mmcrfs** and **mmchfs** commands.

noatime
> Do not update inode access times on this file system. This option can also be controlled with the **-S** option on the **mmcrfs** and **mmchfs** commands.

nomtime
> Update file modification times only periodically. This option can also be controlled with the **-E** option on the **mmcrfs** and **mmchfs** commands.

norelatime
> Update inode access time for each access. This option is the default if `minReleaseLevel` is less than 5.0.0 when the file system is created. This option can also be controlled with the **-S** option of the **mmcrfs** and **mmchfs** commands. For more information, see the topic *atime values* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

nosyncnfs
> Do not commit metadata changes coming from the NFS daemon synchronously. Normal file system synchronization semantics apply. On AIX nodes, **nosyncnfs** is the default. On Linux nodes, **syncnfs** is the default.

relatime
> Allow the update of inode access time only if one of the following is true:

- The existing access time is older than 24 hours. Access time is user configurable through the **atimeDeferredSeconds** configuration attribute.
- The existing file modification time is greater than the existing access time.

This option is the default if `minReleaseLevel` is 5.0.0 or greater when the file system is created. This option can also be controlled with the **-S** option of the **mmcrfs** and **mmchfs** commands. For more information, see the topic *atime values* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

`syncnfs`
   Synchronously commit metadata changes coming from the NFS daemon. On Linux nodes, **syncnfs** is the default. On AIX nodes, **nosyncnfs** is the default.

`useNSDserver={always | asfound | asneeded | never}`
   Controls the initial disk discovery and failover semantics for NSD disks. The possible values are:

   `always`
      Always access the disk using the NSD server. Local dynamic disk discovery is disabled.

   `asfound`
      Access the disk as found (the first time the disk was accessed). No change of disk access from local to NSD server, or the other way around, is performed by GPFS.

   `asneeded`
      Access the disk any way possible. This is the default.

   `never`
      Always use local disk access.

Mounting a file system through GUI

You can use the IBM Spectrum Scale GUI to mount or unmount individual file systems or multiple file systems on the selected nodes. Use the **Files** > **File Systems**, **Files** > **File Systems** > **View Details** > **Nodes**, or **Nodes** > **View Details** > **File Systems** page in the GUI to mount or unmount a file system.

The GUI has the following options related to mounting the file system:

1. Mount local file systems on nodes of the local IBM Spectrum Scale cluster.
2. Mount remote file systems on local nodes.
3. Select individual nodes, protocol nodes, or nodes by node class while selecting nodes on which the file system needs to be mounted.
4. Prevent or allow file systems from mounting on individual nodes.

   Do the following to prevent file systems from mounting on a node:

   a. Go to **Nodes** .
   b. Select the node on which you need to prevent or allow file system mounts.
   c. Select **Prevent Mounts** from the **Actions** menu.
   d. Select the required option and click **Prevent Mount** or **Allow Mount** based on the selection.
5. Configure automatic mount option. The automatic configure option determines whether to automatically mount file system on nodes when GPFS daemon starts or when the file system is accessed for the first time. You can also specify whether to exclude individual nodes while enabling the automatic mount option. To enable automatic mount, do the following:

   a. From the **Files** > **File Systems** page, select the file system for which you need to enable automatic mount.
   b. Select **Configure Automatic Mount** option from the **Actions** menu.
   c. Select the required option from the list of automatic mount modes.
   d. Click **Configure**.

Note: You can configure automatic mount option for a file system only if the file system is unmounted from all nodes. That is, you need to stop I/O on this file system to configure this option. However, you can include or exclude the individual nodes for automatic mount without unmounting the file system from all nodes.

Changing a file system mount point on protocol nodes

If required, you can change a file system mount point on IBM Spectrum Scale protocol nodes.

To change a file system mount point on protocol nodes, perform the following steps:

1. Unmount the file system:

   ```
   mmumount fs0 -a
   ```

2. Change the mount point:

   ```
   mmchfs fs0 -T /ibm/new_fs0
   ```

3. Change the path of all NFS and SMB exports.

   Note: The **mmnfs export change** and the **mmsmb export change** commands do not allow path names to be edited. Therefore, the export needs to be removed and re-added.

4. Change the object CCR files:
   * `account-server.conf`
   * `container-server.conf`
   * `object-server-.conf`
   * `object-server-sof.conf`
   * `spectrum-scale-object.comf`
   * `spectrum-scale-objectizer.conf`

     The parameter that you need to change varies depending on the configuration file.

     a. Use the **mmobj config change** command to list the parameters for the file. For example, to list the parameters for the object-server.conf file, enter:

        ```
        mobj config list --ccrfile object-server.conf --section DEFAULT --property devices
        ```

     b. Use the **mmobj config change --ccrfile** *file name* to change the parameter. For example, to change the object-server.conf file, enter:

        ```
        mmobj config change --ccrfile object-server.conf --section DEFAULT --property devices
         /newFS/name
        ```

Unmounting a file system

Some GPFS administration tasks require you to unmount the file system before they can be performed. You can unmount a GPFS file system using the **mmumount** command.

If the file system does not unmount, see the *File system fails to unmount* section in the *IBM Spectrum Scale: Problem Determination Guide*.

To unmount a GPFS file system using the **mmumount** command, enter:

```
mmumount device
```

where *device* is the name of the file system. For example, to unmount the file system **fs1**, enter:

```
mmumount fs1
```

Unmounting a file system on multiple nodes

This topic describes how to unmount a file systems on multiple nodes.

To unmount file system **fs1** on all nodes in the GPFS cluster, issue this command:

```
mmumount fs1 -a
```

To unmount a file system only on a specific set of nodes, use the **-N** flag of the **mmumount** command.

Unmounting a file system through GUI

You can use the IBM Spectrum Scale GUI to mount or unmount individual file systems or multiple file systems on the selected nodes. Use the **Files** > **File Systems**, **Files** > **File Systems** > **View Details** > **Nodes**, or **Nodes** > **View Details** > **File Systems** page in the GUI to mount or unmount a file system.

You can utilize the following unmount features that are supported in the GUI:

1. Unmount local file system from local nodes and remote nodes.
2. Unmount a remote file system from the local nodes. When a local file system is unmounted from the remote nodes, the remote nodes can no longer be seen in the GUI. The **Files** > **File Systems** > **View Details** > **Remote Nodes** page lists the remote nodes that currently mount the selected file system. The selected file system can be a local or a remote file system but the GUI permits to umount only local file systems from the remote nodes.
3. Select individual nodes, protocol nodes, or nodes by node class while selecting nodes from which the file system needs to be unmounted.
4. Specify whether to force unmount. Selecting the **Force unmount option** while unmounting the file system unmounts the file system even if it is still busy in performing the I/O operations. Forcing the unmount operation affects the outstanding operations and causes data integrity issues. The IBM Spectrum Scale system relies on the native unmount command to carry out the unmount operation. The semantics of forced unmount are platform-specific. On certain platforms such as Linux, even when forced unmount is requested, file system cannot be unmounted if it is still referenced by system kernel. To unmount a file system in such cases, identify and stop the processes that are referencing the file system. You can use system utilities like *lsof* and *fuser* for this.

Deleting a file system

Before deleting a file system, unmount it on all nodes.

Specify the file system to be deleted on the **mmdelfs** command. For example, to delete the file system **fs1**, enter:
```
mmdelfs fs1
```

The system displays information similar to:
```
GPFS: 6027-573 All data on the following disks of fs1 will be destroyed:

    gpfs9nsd
    gpfs13nsd
    gpfs11nsd
    gpfs12nsd

GPFS: 6027-574 Completed deletion of file system fs1.
mmdelfs: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

For more information, see the following:

- "Unmounting a file system" on page 112
- **mmdelfs command** in *IBM Spectrum Scale: Command and Programming Reference* for complete usage information
- **mmdelnsd command** in *IBM Spectrum Scale: Command and Programming Reference* for removing the NSD definitions after deleting the file system

Determining which nodes have a file system mounted

The **mmlsmount** command is used to determine which nodes have a given file system mounted. The name and IP address of each node that has the file system mounted is displayed. This command can be used for all file systems, all remotely mounted file systems, or file systems mounted on nodes of certain clusters.

Note that the **mmlsmount -L** command reports file systems that are in use at the time the command is issued. A file system is considered to be in use if it is explicitly mounted with the **mount** or **mmmount** command or if it is mounted internally for the purposes of running some other GPFS command. For example, when you run the **mmrestripefs** command, the file system will be internally mounted for the duration of the command. If **mmlsmount** is issued in the interim, the file system will be reported as being in use by the **mmlsmount** command but, unless it is explicitly mounted, will not show up in the output of the **mount** or **df** commands. For more details, see For more information, see the topic *The mmlsmount command* in the *IBM Spectrum Scale: Problem Determination Guide.*.

This is an example of a **mmlsmount -L** command for a mounted file system named **fs1**:

```
File system fs1 (mnsd.cluster:fs1) is mounted on 5 nodes:
  9.114.132.101   c5n101                         mnsd.cluster
  9.114.132.100   c5n100                         mnsd.cluster
  9.114.132.106   c5n106                         mnsd.cluster
  9.114.132.97    c5n97                          cluster1.cluster
  9.114.132.92    c5n92                          cluster1.cluster
```

Checking and repairing a file system

The **mmfsck** command finds and repairs conditions that can cause problems in your file system. The **mmfsck** command operates in two modes: online and offline.

The online mode operates on a mounted file system and is chosen by issuing the **-o** option. Conversely, the offline mode operates on an unmounted file system. In general, it is unnecessary to run **mmfsck** in offline mode unless under the direction of the IBM Support Center.

The online mode checks and recovers only unallocated blocks on a mounted file system. If a GPFS file operation fails due to an out of space condition, the cause might be disk blocks that are unavailable after repeated node failures. The corrective action that is taken is to mark the block free in the allocation map. Any other inconsistencies that are found are only reported, not repaired.

Note:

1. If you are running the online **mmfsck** command to free allocated blocks that do not belong to any files, plan to make file system repairs when system demand is low. This is I/O intensive activity and it can affect system performance.
2. If you are repairing a file system due to node failure and the file system has quotas that are enabled, it is suggested that you run the **mmcheckquota** command to make quota accounting consistent.

To repair any other inconsistencies, you must run the offline mode of the **mmfsck** command on an unmounted file system. The offline mode checks for these file inconsistencies that might cause problems:

- Blocks marked allocated that do not belong to any file. The corrective action is to mark the block free in the allocation map.
- Files and directories for which an inode is allocated and no directory entry exists, known as orphaned files. The corrective action is to create directory entries for these files in a **lost+found** subdirectory in the root directory of the fileset to which the file or directory belongs. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. The index number of the inode is assigned as the name. If you do not allow the **mmfsck** command to reattach an orphaned file, it asks for permission to delete the file.

- Directory entries that point to an inode that is not allocated. The corrective action is to remove the directory entry.
- Incorrectly formed directory entries. A directory file contains the inode number and the generation number of the file to which it refers. When the generation number in the directory does not match the generation number that is stored in the file's inode, the corrective action is to remove the directory entry.
- Incorrect link counts on files and directories. The corrective action is to update them with accurate counts.
- Policy files that are not valid. The corrective action is to delete the file.
- Various problems that are related to filesets: missing or corrupted fileset metadata, inconsistencies in directory structure related to filesets, missing or corrupted fileset root directory, other problems in internal data structures. The repaired filesets are renamed as **Fileset** *FilesetId* and put into unlinked state.

The **mmfsck** command performs other functions that are not listed here, as deemed necessary by GPFS.

The **--patch-file** parameter of the **mmfsck** command can be used to generate a report of file system inconsistencies. Consider this example of a patch file that is generated by **mmfsck** for a file system with a bad directory inode:

```
gpfs_fsck

<header>
  sgid = "C0A87ADC:5555C87F"
  disk_data_version = 1
  fs_name = "gpfsh0"
  #patch_file_version = 1
  #start_time = "Fri May 15 16:32:58 2015"
  #fs_manager_node = "h0"
  #fsck_flags = 150994957
</header>

<patch_inode>
  patch_type = "dealloc"
  snapshot_id = 0
  inode_number = 50432
</patch_inode>

<patch_block>
  snapshot_id = 0
  inode_number = 3
  block_num = 0
  indirection_level = 0
  generation_number = 1
  is_clone = false
  is_directory_block = true
  rebuild_block = false
  #num_patches = 1

  <patch_dir>
    entry_offset = 48
    entry_fold_value = 306661480
    delete_entry = true
  </patch_dir>
</patch_block>

<patch_block>
  snapshot_id = 0
  inode_number = 0
  block_num = 0
  indirection_level = 0
  generation_number = 4294967295
  is_clone = false
```

```
  is_directory_block = false
  rebuild_block = false
  #num_patches = 1

  <patch_field>
    record_number = 3
    field_id = "inode_num_links"
    new_value = 2
    old_value = 3
  </patch_field>
</patch_block>

<patch_inode>
  patch_type = "orphan"
  snapshot_id = 0
  inode_number = 50433
</patch_inode>

<footer>
  #stop_time = "Fri May 15 16:33:06 2015"
  #num_sections = 203
  #fsck_exit_status = 8
  need_full_fsck_scan = false
</footer>
```

The **mmfsck** command can be run with both the **--patch-file** and **--patch** parameters to repair a file system with the information that is stored in the patch file. Using a patch file prevents a subsequent scan of the file system before the repair actions begin.

You cannot run the **mmfsck** command on a file system that has disks in a **down** state. You must first run the **mmchdisk** command to change the state of the disks to **unrecovered** or **up**. To display the status of the disks in the file system, issue the **mmlsdisk** command.

To check the file system **fs1** without making any changes to the file system, issue the following command:

```
mmfsck fs1
```

For complete usage information, see **mmchdisk command**, **mmcheckquota command**, **mmfsck command**, and **mmlsdisk command** in *IBM Spectrum Scale: Command and Programming Reference*

Dynamic validation of descriptors on disk

IBM Spectrum Scale can periodically scan descriptors on disk to detect and fix corruption early rather than waiting until the next remount.

The first time a file system gets mounted, a periodic validation of the nsd, disk, and stripe group descriptors gets started. This validation occurs, by default, every five seconds. The nsd, disk, and stripe group descriptors are read and compared with the corresponding descriptors in memory or cache. If there is a mismatch, that information is logged and, if appropriate, the corrupted data is fixed using data from cache.

Listing file system attributes

Use the **mmlsfs** command to display the current file system attributes. Depending on your configuration, additional information that is set by GPFS can be displayed to help in problem determination when you contact the IBM Support Center.

If you specify no options with the **mmlsfs** command, all file system attributes are listed.

For example, to list all of the attributes for the file system gpfs1, enter:

```
mmlsfs gpfs1
```

The system displays information similar to the following:

```
 flag                value                    description
------------------- ------------------------ ------------------------------------
 -f                  8192                     Minimum fragment (subblock) size in bytes
 -i                  4096                     Inode size in bytes
 -I                  16384                    Indirect block size in bytes
 -m                  2                        Default number of metadata replicas
 -M                  2                        Maximum number of metadata replicas
 -r                  2                        Default number of data replicas
 -R                  2                        Maximum number of data replicas
 -j                  cluster                  Block allocation type
 -D                  nfs4                     File locking semantics in effect
 -k                  all                      ACL semantics in effect
 -n                  32                       Estimated number of nodes that will mount file system
 -B                  262144                   Block size
 -Q                  user;group;fileset       Quotas accounting enabled
                     user;group;fileset       Quotas enforced
                     none                     Default quotas enabled
 --perfileset-quota  No                       Per-fileset quota enforcement
 --filesetdf         No                       Fileset df enabled?
 -V                  18.00 (5.0.0.0)          File system version
 --create-time       Mon Aug 28 20:21:17 2017 File system creation time
 -z                  No                       Is DMAPI enabled?
 -L                  134217728                Logfile size
 -E                  Yes                      Exact mtime mount option
 -S                  No                       Suppress atime mount option
 -K                  whenpossible             Strict replica allocation option
 --fastea            Yes                      Fast external attributes enabled?
 --encryption        No                       Encryption enabled?
 --inode-limit       607488                   Maximum number of inodes in all inode spaces
 --log-replicas      2                        Number of log replicas
 --is4KAligned       yes                      is4KAligned?
 --rapid-repair      yes                      rapidRepair enabled?
 --write-cache-threshold 65536                HAWC Threshold (max 65536)
 --subblocks-per-full-block 32                Number of subblocks per full block
 -P                  system                   Disk storage pools in file system
 --file-audit-log    No                       File Audit Logging enabled?
 -d                  nsd20;nsd21;nsd3         Disks in file system
 -A                  yes                      Automatic mount option
 -o                  none                     Additional mount options
 -T                  /gpfs1                   Default mount point
 --mount-priority    0                        Mount priority
```

Some of the attributes that are displayed by the **mmlsfs** command represent default mount options. Because the scope of mount options is an individual node, it is possible to have different values on different nodes. For exact **mtime** (**-E** option) and suppressed **atime** (**-S** option), the information that is displayed by the **mmlsfs** command represents the current setting on the file system manager node. If these options are changed with the **mmchfs** command, the change might not be reflected until the file system is remounted.

For complete usage information, see `mmlsfs command` in *IBM Spectrum Scale: Command and Programming Reference*. For a detailed discussion of file system attributes, see *GPFS architecture* and *File system creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Modifying file system attributes

Use the **mmchfs** command to modify existing file system attributes.

Note: All files created after issuing the **mmchfs** command take on the new attributes. Existing files are not affected. Use the **mmchattr** or **mmrestripefs -R** command to change the replication factor of existing files. See "Querying and changing file replication attributes."

For example, to change the default data replication factor to 2 for the file system **fs1**, enter:

```
mmchfs fs1 -r 2
```

To confirm the changes, enter:

```
mmlsfs fs1 -r
```

The system displays information similar to:

```
flag                value                   description
------------------- ----------------------- ----------------------------------
 -r                 2                       Default number of data replicas
```

For complete usage information, see **mmchfs command** and **mmlsfs command** in *IBM Spectrum Scale: Command and Programming Reference*. For a detailed discussion of file system attributes, see *GPFS architecture* and *File system creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Querying and changing file replication attributes

If your availability requirements change, you can have GPFS display the current replication factors for one or more files by issuing the **mmlsattr** command. You might then decide to change replication for one or more files using the **mmchattr** command.

For complete usage information, see **mmlsattr command** and **mmchattr command** in *IBM Spectrum Scale: Command and Programming Reference*.

Querying file replication

Specify one or more file names with the **mmlsattr** command.

For example, to display the replication factors for two files named **project4.sched** and **project4.resource** in the file system **fs1**, enter:

```
mmlsattr /fs1/project4.sched /fs1/project4.resource
```

The system displays information similar to:

```
  replication factors
metadata(max) data(max) file     [flags]
------------- --------- ---------------
     1 (  2)   1 (  2) /fs1/project4.sched
     1 (  2)   1 (  2) /fs1/project4.resource
```

See the **mmlsattr command** in *IBM Spectrum Scale: Command and Programming Reference* for complete usage information. For a detailed discussion of file system attributes, see *GPFS architecture* and *File system creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Changing file replication attributes

Use the **mmchattr** command to change the replication attributes for one or more files.

You can only increase data and metadata replication as high as the maximum data and maximum metadata replication factors for that file system. You cannot change the maximum data and maximum metadata replication factors once the file system has been created.

Specify the file name, attribute, and new value with the **mmchattr** command. For example, to change the metadata replication factor to 2 and the data replication factor to 2 for the file named **project7.resource** in the file system **fs1**, enter:

```
mmchattr -m 2 -r 2 /fs1/project7.resource
```

To confirm the change, enter:

```
mmlsattr /fs1/project7.resource
```

The system displays information similar to:

```
replication factors
metadata(max) data(max) file     [flags]
------------- --------- ---------------
    2 ( 2)   2 ( 2)   /fs1/project7.resource
```

See the `mmchattr command` and the `mmlsattr command` in *IBM Spectrum Scale: Command and Programming Reference* for complete usage information. For a detailed discussion of file system attributes, see *GPFS architecture* and *File system creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Using Direct I/O on a file in a GPFS file system

The Direct I/O caching policy can be set for files in a GPFS file system by specifying the **-D** option on the **mmchattr** command.

This caching policy bypasses file cache and transfers data directly from disk into the user space buffer, as opposed to using the normal cache policy of placing pages in kernel memory. Applications with poor cache hit rates or very large I/Os may benefit from the use of Direct I/O.

Direct I/O may also be specified by supplying the **O_DIRECT** file access mode on the **open()** of the file.

File compression

You can compress or decompress files either with the `mmchattr` command or with the `mmapplypolicy` command with a `MIGRATE` rule. You can do the compression or decompression synchronously or defer it until a later call to `mmrestripefile` or `mmrestripefs`.

Beginning with IBM Spectrum Scale version 5.0.0, file compression supports two compression libraries, zlib and lz4. Zlib is intended primarily for cold data and favors saving space over read-access speed. Lz4 is intended primarily for active data and favors read-access speed over maximized space saving. Administrators can create policies that select a compression library based on the access characteristics of the file to be compressed, with file-level granularity.

Files that belong to AFM and AFM DR filesets can also be compressed and decompressed. Compressed file contents are decompressed before being transferred from home to cache or from primary to secondary.

For more information about file compression, see the following subtopics:
- "Comparison with object compression" on page 120
- "When to use file compression" on page 120
- "Setting up file compression and decompression" on page 120
- "Warning" on page 121
- "Reported size of compressed files" on page 121
- "Deferred file compression" on page 121
- "Indicators of file compression or decompression" on page 122

- "Partially compressed files" on page 123
- "Updates to compressed files" on page 123
- "File compression and memory mapping" on page 124
- "File compression and direct I/O" on page 124
- "Backing up and restoring compressed files" on page 124
- "FPO environment" on page 124
- "Limitations" on page 125

Comparison with object compression

File compression and object compression use the same compression technology but are available in different environments and are configured in different ways. Object compression is available in the Cluster Export Systems (CES) environment and is configured with the `mmobj policy` command. With object compression, you can create an object storage policy that periodically compresses new objects and files in a GPFS fileset.

File compression is available in non-CES environments and is configured with the `mmapplypolicy` command or directly with the `mmchattr` command.

When to use file compression

File compression supports two compression libraries, zlib and lz4.
- Zlib is intended primarily for cold objects and files. It favors saving space over read-access speed. Compressing other types of data can result in performance degradation.
- Lz4 is intended primarily for active data and favors read-access speed over maximized space saving.

Setting up file compression and decompression

The sample script `/usr/lpp/mmfs/samples/ilm/mmcompress.sample`, installed with IBM Spectrum Scale, provides examples of how to compress or decompress a fileset or a directory tree.

You can do file compression or decompression with either the `mmchattr` command or the `mmapplypolicy` command.

Note: File compression and decompression with the `mmapplypolicy` command is not supported on Windows.

With the `mmchattr` command, you specify the `--compression` option and the names of the files or filesets that you want to compress or decompress. For example, the following command compresses a file with the lz4 compression library:

```
mmchattr --compression lz4 trcrpt.150913.13.30.13.3518.txt
```

The following command decompresses the same file:

```
mmchattr --compression no trcrpt.150913.13.30.13.3518.txt
```

For more information, see the topic *mmchattr command* in the *IBM Spectrum Scale: Command and Programming Reference*.

With the `mmapplypolicy` command, you create a `MIGRATE` rule that specifies the `COMPRESS` option and run `mmapplypolicy` to apply the rule. For example, the following rule selects files with names that contain the string green from the datapool storage pool and compresses them with the zlib library:

```
RULE 'COMPR1' MIGRATE FROM POOL 'datapool' COMPRESS('z') WHERE NAME LIKE 'green%'
```

The following rule decompresses the same set of files:

```
RULE 'COMPR1' MIGRATE FROM POOL 'datapool' COMPRESS('no') WHERE NAME LIKE 'green%'
```

The following example shows three rules:
- The first rule excludes from compression any file that ends with `.mpg` or `.jpg`.
- The second rule automatically compresses any file that was not accessed in the last 30 days with zlib (`libz.so`).
- The third rule automatically compresses any file that was not modified in the last 2 days with lz4 (`liblz4.so`).

```
RULE 'NEVER_COMPRESS' EXCLUDE WHERE lower(NAME) LIKE '%.mpg' OR lower(NAME) LIKE '%.jpg'
RULE 'COMPRESS_COLD' MIGRATE COMPRESS('z') WHERE (CURRENT_TIMESTAMP - ACCESS_TIME) >
(INTERVAL '30' DAYS)
RULE 'COMPRESS_ACTIVE' MIGRATE COMPRESS('lz4') WHERE (CURRENT_TIMESTAMP - MODIFICATION_TIME) >
(INTERVAL '2' DAYS) AND (CURRENT_TIMESTAMP - ACCESS_TIME) <= (INTERVAL '30' DAYS)
```

For more information, see the following help topics:
- The topic *mmchattr command* in the *IBM Spectrum Scale: Command and Programming Reference*
- "Overview of policies" on page 362
- "Policy rules: Syntax" on page 364
- "Policy rules: Terms" on page 366

When you do file compression, you can defer the compression operation to a later time. For more information, see the subtopic "Deferred file compression."

Warning

Doing any of the following operations while the **mmrestorefs** command is running can corrupt file data:
- Doing file compression or decompression. This includes compression or decompression with the **mmchattr** command or with a policy and the **mmapplypolicy** command.
- Running the **mmrestripefile** command or the **mmrestripefs** command. Do not run either of these commands for any reason. Do not run these commands to complete a deferred file compression or decompression.

Reported size of compressed files

After a file is compressed, operating system commands, such as `ls -l`, display the uncompressed size. Use du or the GPFS command **mmdf** to display the actual, compressed size. You can also make the **stat()** system call to find how many blocks the file occupies.

Deferred file compression

By default, the command that launches a file compression or decompression does not return until after the compression or decompression operation is completed. However, with both the **mmchattr** command and the **mmapplypolicy** compression, you can defer the compression or decompression operation and have the command return as soon as it completes any other operations. By deferring compression or decompression, you can complete the operation later when the system is not heavily loaded with processes or I/O.

To defer the compression, with either command, specify the **-I defer** option. For example, the following command marks the specified file as needing compression but defers the compression operation:

```
mmchattr -I defer --compression yes trcrpt.150913.13.30.13.3518.txt
```

With the **mmapplypolicy** command, the **-I defer** option defers compression or decompression and data movement or deletion. For example, the following command applies the rules in the file `policyfile` but defers the file operations that are specified in the rules, including compression or decompression:

```
mmapplypolicy fs1 -P policyfile -I defer
```

To complete a deferred compression or decompression, run the **mmrestripefile** command or the **mmrestripefs** command with the **-z** option. (Do not run either of these commands if an **mmrestorefs** command is running. See the warnings in the preceding subtopic "Warning" on page 121.) The following command completes the deferred compression or decompression of the specified file:

```
mmrestripefile -z trcrpt.150913.13.30.13.3518.txt
```

Indicators of file compression or decompression

The **mmlsattr** command displays two indicators that together describe the state of compression or decompression of the specified file:

COMPRESSION
> The **mmlsattr** command displays the `COMPRESSION` flag on the `Misc attributes` line of its output. The flag is followed in parentheses by the name of the compression library that was used to compress the file. See the example of **mmlsattr** output in Figure 3 on page 123. If present, the `COMPRESSION` flag indicates that the file is compressed or is marked for deferred compression. If absent, the absence indicates that the file is uncompressed or is marked for deferred decompression.
>
> **Note:** This flag reflects the state of the GPFS_IWINFLAG_COMPRESSED flag in the **gpfs_iattr64_t** structure of the inode of the file. For more information about this structure, see the topic *gpfs_iattr64_t_structure* in the *IBM Spectrum Scale: Command and Programming Reference*.

illCompressed
> The **mmlsattr** command displays the `illCompressed` flag on the `flags` line of its output. See Figure 3 on page 123. If present, `illCompressed` indicates that the file is marked for compression or decompression but that compression or decompression is not completed. If absent, the absence indicates that compression or decompression is completed. For more information about this structure, see the topic *gpfs_iattr64_t_structure* in the *IBM Spectrum Scale: Command and Programming Reference*.
>
> **Note:**
> - This flag reflects the state of the GPFS_IAFLAG_ILLCOMPRESSED flag in the **gpfs_iattr64_t** structure of the inode of the file. For more information about this structure, see the topic *gpfs_iattr64_t_structure* in the *IBM Spectrum Scale: Command and Programming Reference*.
> - Some file system events can cause the `illCompressed` flag to be set. Consider the following examples:
>   - When data is written into an already compressed file, the existing data remains compressed but the new data is uncompressed. The `illCompressed` flag is set for this file.
>   - When a compressed file is memory-mapped, the memory-mapped area of the file is decompressed before it is read into memory. The `illCompressed` flag is set for this file.
>
> For more information, see the subtopic "Updates to compressed files" on page 123.

In the following example, the output from the **mmlsattr** command includes both the `COMPRESSION` flag and the `illCompressed` flag. This combination indicates that the file is marked for compression but that compression is not completed:

```
mmlsattr -L green02.51422500687
file name:            green02.51422500687
metadata replication: 1 max 2
data replication:     2 max 2
immutable:            no
appendOnly:           no
flags:                illCompressed
storage pool name:    datapool
fileset name:         root
snapshot name:
creation time:        Wed Jan 28 19:05:45 2015
Misc attributes:      ARCHIVE COMPRESSION (library lz4)
Encrypted:            no
```

Figure 3. Compression and decompression flags

Together the `COMPRESSION` and `illCompressed` flags indicate the compressed or uncompressed state of the file. See the following table:

Table 16. COMPRESSION and illCompressed flags

State of the file	COMPRESSION is displayed?	illCompressed is displayed?
Uncompressed.	No	No
Decompression is not complete.	No	Yes
Compressed.	Yes	No
Compression is not complete.	Yes	Yes

Partially compressed files

The `COMPRESSION` flag of a file is set when the user selects the file to be compressed by the **mmchattr** **--compress yes** command or by a policy run. The flag indicates that the user wants the file to be compressed.

If the user specifies the **-I defer** command option with the **mmchattr** command or with a policy run, the `illCompressed` flag of the file is set during the command execution or policy run. The `illCompressed` flag indicates that the request to compress the file has not been fulfilled. The `illCompressed` flag is reset at the conclusion of the actual compression execution of the file, after the **mmrestripefs -z** or **mmrestripefile** **-z** command finishes compressing the file. The `illCompressed` flag can be set again upon updates of the contents of the file that cause update-driven decompression.

The compressibility of a file can change over time if its contents are changed. Different parts of a file may have different compressibility. Based on the 10% space-saving criterion (see the subtopic "Limitations" on page 125), some compression groups (in granularity of 10 data blocks) of a file might be compressed while others are not.

In sum, the state of the `COMPRESSION` flag, on or off, indicates the intention of the user to compress the file or not. The `illCompressed` flag indicates the compression execution status. The actual compression status of the data blocks depends on the `illCompressed` and `COMPRESSION` flags and the compressibility of the current data.

Updates to compressed files

When a compressed file is updated by a write operation, the file system automatically decompresses the region of the file that contains the affected data and sets the `illCompressed` flag. The file system then makes the update. To recompress the file, run the **mmrestripefile** command with the **-z** option, as in the following example:

```
mmrestripefile -z trcrpt.150913.13.30.13.3518.txt
```

The **mmrestorefs** command can cause a compressed file in the active file system to become decompressed if it is overwritten by the restore process. To recompress the file, run the **mmrestripefile** command with the **-z** option.

For more information, see the preceding subtopic "Deferred file compression" on page 121.

File compression and memory mapping

You can memory-map a file that is already compressed. The file system automatically decompresses the paged-in region and sets the `illCompressed` flag. To recompress the file, run the **mmrestripefile** command with the **-z** option.

As a convenience, the file system does not compress an uncompressed file or partially decompressed file if the file is memory-mapped. Compressing the file would not be not effective because memory mapping decompresses any compressed data in the regions that are paged in.

File compression and direct I/O

You can open a compressed file for Direct I/O, but internally the direct I/O reads and writes are replaced by buffered decompressed I/O reads and writes.

As a convenience, the file system does not compress a file that is opened for Direct I/O. Compressing the file would not be effective because direct I/O would be replaced by buffered decompressed I/O.

Backing up and restoring compressed files

Files are decompressed when they are moved out of storage that is directly managed by IBM Spectrum Scale. This fact affects file backups by products such as IBM Spectrum Protect, IBM Spectrum Protect for Space Management (HSM), IBM Spectrum Archive™, Transparent cloud tiering (TCT), and others. When you back up a file with these products, the file system decompresses the file data inline when it is read by the backup agent. The file system also sets the `illCompressed` flag in the file properties. The backed-up file data is not compressed.

When you restore a file to the IBM Spectrum Scale file system, the file data remains uncompressed but the `illCompressed` flag is still set. You can recompress the file by running **mmrestripefs** or **mmrestripefile** with the **-z** option.

FPO environment

File compression supports a File Placement Optimizer (FPO) environment or horizontal storage pools.

FPO block group factor: Before you compress files in a File Placement Optimizer (FPO) environment, you must set the block group factor to a multiple of 10. If you do not, then data block locality is not preserved and performance is slower.

For compatibility reasons, before you do file compression with FPO files, you must upgrade the whole cluster to version 4.2.1 or later. To verify that the cluster is upgraded, follow these steps:
1. At the command line, enter the **mmlsconfig** command with no parameters.
2. In the output, verify that `minReleaseLevel` is 4.2.1 or later.

AFM environment

Files that belong to AFM and AFM DR filesets can also be compressed and decompressed. Compressed file contents are decompressed before being transferred from home to cache or from primary to secondary.

| Before you do file compression with AFM and AFM DR, you must upgrade the whole cluster to version
| 5.0.0.

Limitations

See the restrictions that are stated in the following subtopics:
- "File compression and memory mapping" on page 124
- "File compression and direct I/O" on page 124
- "Backing up and restoring compressed files" on page 124

File compression has the following limitations:
- File compression processes each compression group within a file independently. A compression group consists of one to 10 consecutive data blocks within a file. If the file contains fewer than 10 data blocks, the whole file is one compression group. If the saving of space for a compression group is less than 10%, file compression does not compress it but skips to the next compression group.
- For file-enabled compression in an FPO-enabled file system, the block group factor must be a multiple of 10 so that the compressed data maintains data locality. If the block group factor is not a multiple of 10, the data locality is broken.
- Direct I/O is not supported for compressed files.
- The following operations are not supported:
  - Compressing files in snapshots
  - Compressing a clone
  - Compressing small files (files that occupy fewer than two subblocks, compressing small files into an inode).
  - Compressing files other than regular files, such as directories.
  - Cloning a compressed file
  - Compressing an open file that is memory-mapped. See the subtopic "File compression and memory mapping" on page 124.
- On Windows:
  - Compression or decompression with the `mmapplypolicy` command is not supported.
  - Compression of files in Windows hyper allocation mode is not supported.
  - The following Windows APIs are not supported:
    - FSCTL_SET_COMPRESSION to enable/disable compression on a file
    - FSCTL_GET_COMPRESSION to retrieve compression status of a file.
  - In Windows Explorer, in the Advanced Attributes window, the compression feature is not supported.

Setting the Quality of Service for I/O operations (QoS)

QoS limits the effect of I/O-intensive GPFS maintenance commands on overall system I/O performance.

With QoS, you can prevent I/O-intensive, long-running GPFS maintenance commands from dominating file system I/O performance and significantly delaying other tasks. Commands like the examples in Figure 4 can generate hundreds or thousands of requests for I/O operations per second. The high demand can greatly slow down normal tasks that are competing for the same I/O resources.

```
mmrestripefs fsname -N
mmapplypolicy fsname -N all ...
```

Figure 4. Examples of long-running, IO-intensive GPFS commands

The I/O intensive, potentially long-running GPFS commands are collectively called *maintenance commands* and are listed in the help topic for the *mmchqos command* in the *IBM Spectrum Scale: Command and Programming Reference*.

With QoS configured, you can assign an instance of a maintenance command to a QoS class that has a lower I/O priority. Although the instance now takes longer to run to completion, normal tasks have greater access to I/O resources and run more quickly.

For more information, see the descriptions of the QoS commands:
- *mmchqos command* in the *IBM Spectrum Scale: Command and Programming Reference*
- *mmlsqos command* in the *IBM Spectrum Scale: Command and Programming Reference*

Note:
- QoS requires the file system to be at V4.2.0.0 or later. To check the file system level, enter the following command:

  ```
  mmlsfs fileSystemName -V
  ```
- QoS works with asynchronous I/O, memory-mapped I/O, cached I/O, and buffered I/O. However, with direct I/O, QoS counts the IOPS but does not regulate them.

Overview of using QoS

The following steps provide an overview of how to use QoS. In this overview, assume that the file system fs0 contains 5 nodes and has two storage pools: the system storage pool (system) and another storage pool sp1.

1. Monitor your file system with the **mmlsqos** command to determine its maximum capacity in I/O operations per second (IOPS). Follow these steps:

   a. Enable QoS without placing any limits on I/O consumption. The following command sets the QoS classes of both storage pools to **unlimited**:

Table 17. Set QoS classes to **unlimited**

Storage pool	QoS class: maintenance	QoS class: other
system	unlimited	unlimited
sp1	unlimited	unlimited

   ```
   mmchqos fs0 --enable --reset
   ```
   b. Run some maintenance commands that drive I/O on all nodes and disks.
   c. Run the **mmlsqos** command to observe how many IOPS are consumed:

   ```
   mmlsqos fs0 --seconds 60
   ```
2. Run the **mmchqos** command to allocate the available IOPS among the storage pools.

   a. Allocate a smaller share of IOPS to the **maintenance** class, perhaps 15 percent. For example, if you determined in Step 1 that the maximum is 10,000 IOPS, then you might allocate 1500 IOPS to the maintenance class.

   If there is more than one storage pool, then divide the IOPS among the maintenance classes of the storage pools. In this overview, suppose that you decide to allocate 1000 IOPS to the maintenance class of the system pool and 500 IOPS to the maintenance class of the sp1 storage pool. See the second column of the table below.

   Note: Make sure that the virtual storage Logical Unit Numbers (LUNs) of different storage pools do not map to the same physical devices.

   By default, QoS divides specific allocations of IOPS evenly among the nodes in the file system. In this overview there are 5 nodes. So QoS allocates 200 IOPS to the **maintenance** class of the system pool and 100 IOPS to the **maintenance** class of the sp1 storage pool on each node.

Note: You can also divide IOPS among a list of nodes or among the nodes of a node class. For example, you can use the **mmcrnodeclass** command to create a class of nodes that do maintenance commands. You can than divide IOPS among the members of the node class by entering a command like the following one:

```
mmchqos fs0 --enable -N nodeClass pool=sp2,maintenance=880IOPS,other=unlimited
```

If the file system serves remote clusters, you can divide IOPS among the members of a remote cluster by entering a command like the following one:

```
mmchqos fs0 --enable -C remoteCluster pool=sp3,maintenance=1000IOPS,other=unlimited
```

   b. Allocate the remaining IOPS to the **other** classes. It is a good idea to accomplish this task by setting **other** to **unlimited** in each storage class. Then normal tasks can absorb all the IOPS of the system when no maintenance commands are running. See the third column of the following table:

Table 18. Allocate the available IOPS

Storage pool	QoS class: maintenance	QoS class: other
system	1000 IOPS (200 IOPS per node)	**unlimited**
sp1	500 IOPS (100 IOPS per node)	**unlimited**

The command is on one line:

```
mmchqos fs0 --enable pool=system,maintenance=1000IOPS,other=unlimited
 pool=sp1,maintenance=500IOPS,other=unlimited
```

3. When you run a maintenance command, QoS by default assigns it to the **maintenance** class:

```
mmdeldisk fs0 nsd12
```

All maintenance command instances that are running at the same time and that access the same storage pool compete for the IOPS that you allocated to the maintenance class of that storage pool. If the IOPS limit of the class is exceeded, then QoS queues the extra I/O requests until more IOPS become available.

To run a maintenance command without I/O restrictions, you can explicitly assign it to the **other** class:

```
mmdeldisk fs0 nsd12 --qos other
```

4. You can disable QoS at any time without losing your IOPS allocations:

```
mmchqos fs0 --disable
```

When you reenable QoS it starts applying the allocations again:

```
mmchqos fs0 --enable
```

5. You can change the IOPS allocations at any time. The following command is on one line:

```
mmchqos fs0 --enable pool=system,maintenance=750IOPS,other=unlimited
 pool=sp1,maintenance=750IOPS,other=unlimited
```

When you change allocations, mount the file system, or reenable QoS, a brief delay due to reconfiguration occurs before QoS starts applying allocations.

6. To monitor the consumption of IOPS while a maintenance command is running, run the **mmlsqos** command. The following command displays the statistics for the preceding 60 seconds during which a maintenance command was running:

```
mmlsqos fs0 --seconds 60
```

See also

- *mmchqos command* in the *IBM Spectrum Scale: Command and Programming Reference*
- *mmlsqos command* in the *IBM Spectrum Scale: Command and Programming Reference*

Restriping a GPFS file system

Writing data into a GPFS file system correctly stripes the file. However, if you have added disks to a GPFS file system that are seldom updated, use the **mmrestripefs** command to restripe the file system to achieve maximum performance. You can also use **mmrestripefs** to perform any incomplete or deferred file compression or decompression.

Restriping offers the opportunity to specify useful options in addition to rebalancing (**-b** option). Re-replicating (**-r** or **-R** option) provides for proper replication of all data and metadata. If you use replication, this option is useful to protect against additional failures after losing a disk. For example, if you use a replication factor of 2 and one of your disks fails, only a single copy of the data would remain. If another disk then failed before the first failed disk was replaced, some data might be lost. If you expect delays in replacing the failed disk, you could protect against data loss by suspending the failed disk using the **mmchdisk** command and re-replicating. This would assure that all data existed in two copies on operational disks.

If files are assigned to one storage pool, but with data in a different pool, the placement (**-p**) option will migrate their data to the correct pool. Such files are referred to as ill-placed. Utilities, such as the **mmchattr** command or policy engine, may change a file's storage pool assignment, but not move the data. The **mmrestripefs** command may then be invoked to migrate all of the data at once, rather than migrating each file individually. Note that the rebalance (**-b**) option also performs data placement on all files, whereas the placement (**-p**) option rebalances only the files that it moves.

If you do not replicate all of your files, the migrate (**-m**) option is useful to protect against data loss when you have an advance warning that a disk may be about to fail, for example, when the error logs show an excessive number of I/O errors on a disk. Suspending the disk and issuing the **mmrestripefs** command with the **-m** option is the quickest way to migrate only the data that would be lost if the disk failed.

If you do not use replication, the **-m** and **-r** options are equivalent; their behavior differs only on replicated files. After a successful re-replicate (**-r** option) all suspended disks are empty. A migrate operation, using the **-m** option, leaves data on a suspended disk as long as at least one other replica of the data remains on a disk that is not suspended. Restriping a file system includes re-replicating it; the **-b** option performs all the operations of the **-m** and **-r** options.

Use the **-z** option to perform any deferred or incomplete compression or decompression of files in the file system.

Consider the necessity of restriping and the current demands on the system. New data which is added to the file system is correctly striped. Restriping a large file system requires extensive data copying and may affect system performance. Plan to perform this task when system demand is low.

If you are sure you want to proceed with the restripe operation:
1. Use the **mmchdisk** command to suspend any disks to which you *do not* want the file system restriped. You may want to exclude disks from file system restriping because they are failing. See "Changing GPFS disk states and parameters" on page 163.
2. Use the **mmlsdisk** command to assure that all disk devices to which you *do* want the file system restriped are in the up/normal state. See "Displaying GPFS disk states" on page 162.

Specify the target file system with the **mmrestripefs** command. For example, to rebalance (**-b** option) file system **fs1** after adding an additional RAID device, enter:
```
mmrestripefs fs1 -b
```

The system displays information similar to:

```
Scanning file system metadata, phase 1 ...
  19 % complete on Wed Mar 14 21:28:46 2012
 100 % complete on Wed Mar 14 21:28:48 2012
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scanning file system metadata for sp1 storage pool
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Wed Mar 14 21:28:55 2012
Scan completed successfully.
```

Note: Rebalancing of files is an I/O-intensive and time-consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

For complete usage information, see **mmrestripefs command** in *IBM Spectrum Scale: Command and Programming Reference*.

Querying file system space

Although you can use the **df** command to summarize the amount of free space on all GPFS disks, the **mmdf** command is useful for determining how well-balanced the file system is across your disks. (Also, the output from **mmdf** can be more up to date than the output from **df**.) Additionally, you can use the **mmdf** command to diagnose space problems that might result from fragmentation.

Note: The **mmdf** command may require considerable metadata I/O, and should be run when the system load is light.

Specify the file system you want to query with the **mmdf** command. For example, to query available space on all disks in the file system **fs1**, enter:
```
mmdf fs1
```

The system displays information similar to:
```
disk                disk size  failure holds   holds             free KB             free KB
name                   in KB    group metadata data     in full blocks      in fragments
--------------- ------------- -------- -------- ----- -------------------- --------------------
Disks in storage pool: system (Maximum disk size allowed is 122 GB)
hd16vsdn10          17793024       -1 yes      yes      17538560 ( 99%)        1728 ( 0%)
hd3vsdn01            8880128        2 yes      yes       8658176 ( 98%)        1600 ( 0%)
hd4vsdn01            8880128        2 yes      yes       8616448 ( 97%)        1384 ( 0%)
hd15vsdn10          17793024       10 yes      yes      17539584 ( 99%)        1664 ( 0%)
hd13vsdn02           8880128     4001 yes      yes       8663552 ( 98%)        1776 ( 0%)
hd8vsdn01            8880128     4002 yes      yes       8659200 ( 98%)        1936 ( 0%)
hd5vsdn01            8880128     4002 yes      yes       8654848 ( 97%)        1728 ( 0%)
hd33n09             17796008     4003 yes      yes      17540864 ( 99%)        2240 ( 0%)
                -------------                           -------------------- --------------------
(pool total)       257800488                           252091136 ( 98%)       46928 ( 0%)


Disks in storage pool: fs1sp1 (Maximum disk size allowed is 122 GB)
hd30n01              8897968        8 no       yes       8895488 (100%)         424 ( 0%)
hd31n01              8897968        8 no       yes       8895488 (100%)         424 ( 0%)
                -------------                           -------------------- --------------------
(pool total)        17795936                           17790976 (100%)         848 ( 0%)


                =============                         ==================== ====================
(data)             266716296                          261222144 ( 98%)       44576 ( 0%)
(metadata)         248920360                          243217408 ( 98%)       46048 ( 0%)
                =============                         ==================== ====================
```

```
(total)              275596424                    269882112 ( 98%)         47776 ( 0%)

Inode Information
-----------------
Number of used inodes:          9799
Number of free inodes:       4990393
Number of allocated inodes:  5000192
Maximum number of inodes:    5000192
```

For complete usage information, see `mmdf command` in *IBM Spectrum Scale: Command and Programming Reference*.

Querying and reducing file system fragmentation

Disk fragmentation within a file system is an unavoidable condition. When a file is closed after it has been written to, the last logical block of data is reduced to the actual number of subblocks required, thus creating a fragmented block.

In order to **write** to a file system, free full blocks of disk space are required. Due to fragmentation, it is entirely possible to have the situation where the file system is not full, but an insufficient number of free full blocks are available to **write** to the file system. Replication can also cause the copy of the fragment to be distributed among disks in different failure groups. The **mmdefragfs** command can be used to query the current fragmented state of the file system and reduce the fragmentation of the file system.

In order to reduce the fragmentation of a file system, the **mmdefragfs** command migrates fragments to free space in another fragmented disk block of sufficient space, thus creating a free full block. There is no requirement to have a free full block in order to run the **mmdefragfs** command. The execution time of the **mmdefragfs** command depends on the size and allocation pattern of the file system. For a file system with a large number of disks, the **mmdefragfs** command will run through several iterations of its algorithm, each iteration compressing a different set of disks. Execution time is also dependent on how fragmented the file system is. The less fragmented a file system, the shorter time for the **mmdefragfs** command to execute.

The fragmentation of a file system can be reduced on all disks which are not suspended or stopped. If a disk is suspended or stopped, the state of the disk, not the utilization information, will be displayed as output for the **mmdefragfs** command.

The **mmdefragfs** command can be run on both a mounted or an unmounted file system, but achieves best results on an unmounted file system. Running the command on a mounted file system can cause conflicting allocation information and consequent retries to find a new free subblock of the correct size to store the fragment in.

Querying file system fragmentation

To query the current status of the amount of fragmentation for a file system, specify the file system name along with the **-i** option on the **mmdefragfs** command.

For example, to display the current fragmentation information for file system **fs0**, enter:
```
mmdefragfs fs0 -i
```

The system displays information similar to:
```
"fs0"   10304 inodes:    457 allocated / 9847 free

                      free subblk     free
disk           disk size  in full  subblk in      %        %
name           in nSubblk  blocks   fragments free blk blk util
-------------- --------- --------- --------- -------- -------
gpfs68nsd        4390912   4270112       551   97.249   99.544
```

```
gpfs69nsd          4390912   4271360        490   97.277   99.590
                 --------- --------- ---------            -------
(total)            8781824   8541472       1041            99.567
```

For complete usage information, see **mmdefragfs command** in *IBM Spectrum Scale: Command and Programming Reference.*

Reducing file system fragmentation

You can reduce the amount of fragmentation for a file system by issuing the **mmdefragfs** command, with or without a desired block usage goal.

For example, to reduce the amount of fragmentation for file system **fs1** with a goal of 100% utilization, enter:

```
mmdefragfs fs1 -u 100
```

The system displays information similar to:

```
Defragmenting file system 'fs1'...

Defragmenting until full block utilization is 98.00%, currently 97.07%
 27.35 % complete on Tue May 26 14:25:42 2009  (    617882 inodes    4749 MB)
 82.65 % complete on Tue May 26 14:26:02 2009  (   1867101 inodes   10499 MB)
 89.56 % complete on Tue May 26 14:26:23 2009  (   2023206 inodes   14296 MB)
 90.01 % complete on Tue May 26 14:26:43 2009  (   2033337 inodes   17309 MB)
 90.28 % complete on Tue May 26 14:27:03 2009  (   2039551 inodes   19779 MB)
 91.17 % complete on Tue May 26 14:27:23 2009  (   2059629 inodes   23480 MB)
 91.67 % complete on Tue May 26 14:27:43 2009  (   2070865 inodes   26760 MB)
 92.51 % complete on Tue May 26 14:28:03 2009  (   2089804 inodes   29769 MB)
 93.12 % complete on Tue May 26 14:28:23 2009  (   2103697 inodes   32649 MB)
 93.39 % complete on Tue May 26 14:28:43 2009  (   2109629 inodes   34934 MB)
 95.47 % complete on Tue May 26 14:29:04 2009  (   2156805 inodes   36576 MB)
 95.66 % complete on Tue May 26 14:29:24 2009  (   2160915 inodes   38705 MB)
 95.84 % complete on Tue May 26 14:29:44 2009  (   2165146 inodes   40248 MB)
 96.58 % complete on Tue May 26 14:30:04 2009  (   2181719 inodes   41733 MB)
 96.77 % complete on Tue May 26 14:30:24 2009  (   2186053 inodes   43022 MB)
 96.99 % complete on Tue May 26 14:30:44 2009  (   2190955 inodes   43051 MB)
 97.20 % complete on Tue May 26 14:31:04 2009  (   2195726 inodes   43077 MB)
 97.40 % complete on Tue May 26 14:31:24 2009  (   2200378 inodes   43109 MB)
 97.62 % complete on Tue May 26 14:31:44 2009  (   2205201 inodes   43295 MB)
 97.83 % complete on Tue May 26 14:32:05 2009  (   2210003 inodes   43329 MB)
 97.85 % complete on Tue May 26 14:32:25 2009  (   2214741 inodes   43528 MB)
 97.86 % complete on Tue May 26 14:32:55 2009  (   2221888 inodes   43798 MB)
 97.87 % complete on Tue May 26 14:33:35 2009  (   2231453 inodes   44264 MB)
 97.88 % complete on Tue May 26 14:34:26 2009  (   2243181 inodes   45288 MB)
100.00 % complete on Tue May 26 14:35:10 2009

              free subblk            free
disk          in full             subblk in        %             %
name          blocks        blk    fragments     free blk      blk util
           before    after freed  before    after before after before after
---------- ----------------------- ----------------- ----------- -----------
nsd32       277504   287840   323   12931     2183  84.69 87.84  96.05 99.33
nsd33       315232   315456     7     580      185  96.20 96.27  99.82 99.94
nsd21       301824   303616    56    2481      666  92.11 92.66  99.24 99.80
nsd34       275904   285920   313   13598     3159  84.20 87.26  95.85 99.04
nsd30       275840   285856   313   13348     2923  84.18 87.24  95.93 99.11
nsd19       278592   288832   320   12273     1874  85.02 88.14  96.25 99.43
nsd31       276224   284608   262   12012     3146  84.30 86.86  96.33 99.04
           ----------------------- -----------------             -----------
(total)    2001120  2052128  1594   67223    14136             97.07 99.38

Defragmentation complete, full block utilization is 99.04%.
```

See the `mmdefragfs command` in *IBM Spectrum Scale: Command and Programming Reference* for complete usage information.

Protecting data in a file system using backup

GPFS provides ways to back up the file system user data and the overall file system configuration information.

You can use the **mmbackup** command to back up the files of a GPFS file system or the files of an independent fileset to an IBM Spectrum Protect server.

Alternatively, you can utilize the GPFS policy engine (**mmapplypolicy** command) to generate lists of files to be backed up and provide them as input to some other external storage manager.

The file system configuration information can be backed up using the **mmbackupconfig** command.

Note: Windows nodes do not support the **mmbackup**, **mmapplypolicy**, and **mmbackupconfig** commands.

Protecting data in a file system using the mmbackup command

The **mmbackup** command can be used to back up some or all of the files of a GPFS file system to IBM Spectrum Protect servers using the IBM Spectrum Protect Backup-Archive client. After files have been backed up, you can restore them using the interfaces provided by IBM Spectrum Protect.

The **mmbackup** command utilizes all the scalable, parallel processing capabilities of the **mmapplypolicy** command to scan the file system, evaluate the metadata of all the objects in the file system, and determine which files need to be sent to backup in IBM Spectrum Protect, as well which deleted files should be expired from IBM Spectrum Protect. Both backup and expiration take place when running **mmbackup** in the incremental backup mode.

The **mmbackup** command can interoperate with regular IBM Spectrum Protect commands for backup and expire operations. However if after using **mmbackup**, any IBM Spectrum Protect incremental or selective backup or expire commands are used, **mmbackup** needs to be informed of these activities. Use either the **-q** option or the **--rebuild** option in the next **mmbackup** command invocation to enable **mmbackup** to rebuild its shadow databases. (See **mmbackup Examples** in *IBM Spectrum Scale: Command and Programming Reference*.)

These databases *shadow* the inventory of objects in IBM Spectrum Protect so that only new changes will be backed up in the next incremental **mmbackup**. Failing to do so will needlessly back up some files additional times. The shadow database can also become out of date if **mmbackup** fails due to certain IBM Spectrum Protect server problems that prevent **mmbackup** from properly updating its shadow database after a backup. In these cases it is also required to issue the next **mmbackup** command with either the **-q** option or the **--rebuild** options.

The **mmbackup** command provides:
- A full backup of all files in the specified scope.
- An incremental backup of only those files that have changed or been deleted since the last backup. Files that have changed since the last backup are updated and files that have been deleted since the last backup are expired from the IBM Spectrum Protect server.
- Utilization of a fast scan technology for improved performance.
- The ability to perform the backup operation on a number of nodes in parallel.
- Multiple tuning parameters to allow more control over each backup.
- The ability to backup the read/write version of the file system or specific global snapshots.

- Storage of the files in the backup server under their GPFS root directory path independent of whether backing up from a global snapshot or the live file system.
- Handling of unlinked filesets to avoid inadvertent expiration of files.

  Note: Avoid unlinking a fileset while running **mmbackup**. If a fileset is unlinked before **mmbackup** starts, it is handled; however, unlinking a fileset during the job could result in a failure to back up changed files as well as expiration of already backed up files from the unlinked fileset.

The **mmbackup** command supports backing up GPFS file system data to multiple IBM Spectrum Protect servers. The ability to partition file backups across multiple IBM Spectrum Protect servers is particularly useful for installations that have a large number of files. For information on setting up multiple IBM Spectrum Protect servers, see "IBM Spectrum Protect requirements."

Unless otherwise specified, the **mmbackup** command backs up the current active version of the GPFS file system. If you want to create a backup of files at a specific point in time, first use the **mmcrsnapshot** command to create either a global snapshot or a fileset-level snapshot, and then specify that snapshot name for the **mmbackup -S** option. A global snapshot can be specified for either **--scope filesystem** or **--scope inodespace**. A fileset-level snapshot can only be specified with **--scope inodespace**.

If an unlinked fileset is detected, the **mmbackup** processing will issue an error message and exit. You can force the backup operation to proceed by specifying the **mmbackup -f** option. In this case, files that belong to unlinked filesets will not be backed up, but will be removed from the expire list.

If you have file systems that were backed up using the GPFS 3.2 or earlier version of the **mmbackup** command, you will not be able to take advantage of some of the new **mmbackup** features until a new full backup is performed. See "File systems backed up using GPFS 3.2 or earlier versions of mmbackup" on page 135.

Protecting data in a fileset using the `mmbackup` command

The **mmbackup** command can be used to back up an independent fileset to the IBM Spectrum Protect servers using the IBM Spectrum Protect Backup-Archive client. After a fileset has been backed up, you can restore files using the interfaces provided by IBM Spectrum Protect.

When backing up an independent fileset, the **mmbackup** command backs up the current active version of the fileset. The path to the independent fileset root is specified with the *Directory* parameter of the **mmbackup** command.

If you want to create a backup of a fileset at a specific point in time, first use the **mmcrsnapshot** command to create a fileset-level snapshot. Next, specify that snapshot name for the **mmbackup -S** option along with the **--scope inodespace** option.

IBM Spectrum Protect requirements

The **mmbackup** command requires a IBM Spectrum Protect client and server environment to perform a backup operation.

For details on the supported versions of IBM Spectrum Protect, client and server installation and setup, and include and exclude lists, see the IBM Tivoli® Storage Manager V7.1.7 documentation(www.ibm.com/support/knowledgecenter/SSGSG7_7.1.7/com.ibm.itsm.ic.doc/welcome.html).

1. Ensure that the supported versions of the IBM Spectrum Protect client and server are installed. See the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
2. Ensure that the IBM Spectrum Protect server and clients are configured properly for backup operations.
3. If you are using multiple IBM Spectrum Protect servers to protect data, ensure that the IBM Spectrum Protect servers are set up properly.

4. Ensure the required **dsm.sys** and **dsm.opt** configuration files are present in the IBM Spectrum Protect configuration directory on each node used to run **mmbackup** or named in a node specification with **-N**.

5. If you want to include or exclude specific files or directories by using include-exclude lists, ensure that the lists are set up correctly before you invoke the **mmbackup** command.

   The **mmbackup** command uses a IBM Spectrum Protect include-exclude list for including and excluding specific files or directories. See the Tivoli documentation for information about defining an include-exclude list.

   Note: IBM Spectrum Protect interprets its include and exclude statements in a unique manner that is not precisely matched by the GPFS **mmapplypolicy** file selection language. The essential meaning of each supported include or exclude statement is followed, but the commonly used IBM Spectrum Protect idiom of excluding everything as the last statement and including selective directory or file name patterns in prior statements should not be used with GPFS and **mmbackup**. The exclusion pattern of **"/*"** is interpreted by **mmapplypolicy** to exclude everything, and no data is backed up.

   A very large include-exclude list can decrease backup performance. Use wildcards and eliminate unnecessary include statements to keep the list as short as possible.

6. If more than one node will be used to perform the backup operation (**mmbackup -N** option):

   - The **mmbackup** command will verify that the IBM Spectrum Protect Backup-Archive client versions and configuration are correct before executing the backup. Any nodes that are not configured correctly will be removed from the backup operation. Ensure that IBM Spectrum Protect clients are installed and at the same version on all nodes that will invoke the **mmbackup** command or participate in parallel backup operations.

   - Ensure that IBM Spectrum Protect is aware that the various IBM Spectrum Protect clients are all working on the same file system, not different file systems having the same name on different client machines. This is accomplished by using proxy nodes for multiple nodes in the cluster. See the IBM Spectrum Protect documentation for recommended settings for GPFS cluster nodes setup.

7. Restoration of backed-up data must be done using IBM Spectrum Protect interfaces. This can be done with the client command-line interface or the IBM Spectrum Protect web client. The IBM Spectrum Protect web client interface must be made operational if you wish to use this interface for restoring data to the file system from the IBM Spectrum Protect server.

8. When more than one IBM Spectrum Protect server is referenced in the **dsm.sys** file, **mmbackup** uses all listed IBM Spectrum Protect servers by default. To use only a select IBM Spectrum Protect server or the servers that are listed in **dsm.sys**, use the **mmbackup --tsm-servers** option. When more than one IBM Spectrum Protect server is used for backup, the list and the order specified should remain constant. If additional IBM Spectrum Protect servers are added to the backup later, add them to the end of the list that is specified with the **mmbackup --tsm-servers** option.

9. IBM Spectrum Protect does not support special characters in the path names and in some cases cannot back up a path name that has special characters. A limited number of special characters are supported on IBM Spectrum Protect client 6.4.0.0 and later versions with client options **WILDCARDSARELITERAL** and **QUOTESARELITERAL**. Use these IBM Spectrum Protect options with the **mmbackup --noquote** option if you have path names with special characters. The **mmbackup** command does not back up path names containing any newline, **Ctrl+X**, or **Ctrl+Y** characters. If the **mmbackup** command finds unsupported characters in the path name, it writes that path to a file called mmbackup.unsupported.*tsmserver* at the root of the **mmbackup** record directory (by default it is the root of the file system).

Attention: If you are using the IBM Spectrum Protect Backup-Archive client command line or web interface to do back up, use caution when you unlink filesets that contain data backed up by IBM Spectrum Protect. IBM Spectrum Protect tracks files by path name and does not track filesets. As a result, when you unlink a fileset, it appears to IBM Spectrum Protect that you deleted the contents of the fileset. Therefore, the IBM Spectrum Protect Backup-Archive client inactivates the data on the IBM Spectrum Protect server, which may result in the loss of backup data during the expiration process.

File systems backed up using GPFS 3.2 or earlier versions of mmbackup

GPFS 3.2 and earlier versions of the **mmbackup** command automatically created a temporary snapshot named **.mmbuSnapshot** of the specified file system, and backed up this snapshot to the IBM Spectrum Protect server. Accordingly, the files backed up by the command were stored in IBM Spectrum Protect using the */Device*/**.snapshots**/**.mmbuSnapshot** directory path in the remote data store.

The GPFS 3.3 through GPFS 3.5.0.11 versions of the **mmbackup** command will preserve this type of processing for incremental backups until a new full backup is performed. Once a full backup is performed, **mmbackup** will store the files in IBM Spectrum Protect under their usual GPFS root directory path name; all files under */Device*/**.snapshots**/**.mmbuSnapshot** will be marked for expiration. Until the transition to using the usual GPFS root directory path name in IBM Spectrum Protect is complete, no backups can be taken from a snapshot, other than the **mmbackup** temporary snapshot called **.mmbuSnapshot**.

Attention: Starting with GPFS 4.1, the **mmbackup** command will no longer support the */Device*/**.snapshots**/**.mmbuSnapshot** path name format for incremental backups. After migrating to GPFS 4.1, if the older **.mmbuSnapshot** path name format is still in use, a full backup is required if a full backup has never been performed with GPFS 3.3 or later. After the full backup is performed, files will now always be stored in IBM Spectrum Protect under their usual GPFS root directory path name. All files in IBM Spectrum Protect under */Device*/**.snapshots**/**.mmbuSnapshot** will be marked for expiration automatically after a successful backup.

The transition to using the usual GPFS root directory path name format, instead of the */Device*/**.snapshots**/**.mmbuSnapshot** path name format permits **mmbackup** to perform a backup using any user-specified snapshot, or the live file system interchangeably.

Certain features, such as backing up from an arbitrary snapshot, cannot be used until a full backup is performed with the GPFS 3.3 or later version of the **mmbackup** command.

Migrating to mmbackup from IBM Spectrum Protect-interface–based backup

File systems that are backed up using the IBM Spectrum Protect interface can be converted to use the **mmbackup** command to take advantage of the performance offered by **mmbackup** fast scan technology.

A full backup is not required or necessary when moving from backup using the IBM Spectrum Protect interface to the **mmbackup** command.

The **mmbackup** command uses one or more shadow database files to determine changes in the file system. To convert from the IBM Spectrum Protect interface backup to **mmbackup**, one must create the shadow database file or files by using the **--rebuild** option of **mmbackup**. The rebuild option queries the existing IBM Spectrum Protect server or servers and creates a shadow database of the files currently backed up in IBM Spectrum Protect. After the shadow database file or files are generated, **mmbackup** can be used for all future incremental or full backups.

Note: If using multiple IBM Spectrum Protect servers to back up a file system, use the **mmbackup** **--tsm-servers** option to ensure that the proper servers participate in the backup job.

Tuning backups with the `mmbackup` command

You can tune backups with the `mmbackup` command.

The **mmbackup** command performs all its work in three major steps, and all of these steps potentially use multiple nodes and threads:
1. The file system is scanned with **mmapplypolicy**, and a list is created of every file that qualifies and should be in backup for each IBM Spectrum Protect server in use. The existing shadow database and the list generated are then compared and the differences between them yield:
   - Objects deleted recently that should be marked inactive (expire)

- Objects modified or newly created to back up (selective)
- Objects modified without data changes; owner, group, mode, and migration state changes to update (incremental)

2. Using the lists created in step 1 on page 135, **mmapplypolicy** is run for files that should be marked inactive (expire).

3. Using the lists created in step 1 on page 135, **mmapplypolicy** is run for selective or incremental backup.

The **mmbackup** command has several parameters that can be used to tune backup jobs. During the scanning phase, the resources **mmbackup** will utilize on each node specified with the **-N** parameter can be controlled:

- The **-a** *IscanThreads* parameter allows specification of the number of threads and sort pipelines each node will run during the parallel inode scan and policy evaluation. This parameter affects the execution of the high-performance protocol that is used when both the **-g** and **-N** parameters are specified. The default value is 2. Using a moderately larger number can significantly improve performance, but might strain the resources of the node. In some environments a large value for this parameter can lead to a command failure.

  Tip: Set this parameter to the number of CPU *cores* implemented on a typical node in your GPFS cluster.

- The **-n** *DirThreadLevel* parameter allows specification of the number of threads that will be created and dispatched within each **mmapplypolicy** process during the directory scan phase.

During the execution phase for expire, **mmbackup** processing can be adjusted as follows:

- Automatic computation of the ideal expire bunch count. The number of objects named in each file list can be determined, separately from the number in a backup list, and automatically computed, if not specified by the user.
- As an alternative to the automatic computation, the user can control expire processing as follows:
  - The **--max-expire-count** parameter can be used to specify a bunch-count limit for each **dsmc expire** command. This parameter cannot be used in conjunction with **-B**.
  - The **--expire-threads** parameter can be used to control how many threads run on each node running **dsmc expire**. This parameter cannot be used in conjunction with **-m**.

During the execution phase for backup, **mmbackup** processing can be adjusted as follows:

- Automatic computation of ideal backup bunch count. The number of objects named in each file list can be determined, separately from the number in an expire list, and automatically computed, if not specified by the user.
- As an alternative to the automatic computation, the user can control backup processing as follows:
  - The **--max-backup-count** parameter can be used to specify a bunch-count limit for each **dsmc selective** or **dsmc incremental** command. This parameter cannot be used in conjunction with **-B**.
  - The **--backup-threads** parameter can be used to control how many threads run on each node running backup. This parameter cannot be used in conjunction with **-m**.
  - The **--max-backup-size** parameter can be used to further limit the size of a backup bunch by the overall size of all files listed in any single bunch list.

For more information on the **mmbackup** tuning parameters, see `mmbackup command` in *IBM Spectrum Scale: Command and Programming Reference*.

MMBACKUP_PROGRESS_CALLOUT environment variable

The MMBACKUP_PROGRESS_CALLOUT environment variable specifies the path to a program or script to be called during **mmbackup** execution with a formatted argument.

The **$progressCallOut** function is executed if the path **$progressCallOut** names a valid, executable file and one of the following is true:

- The message class provided with this message is 0.

  Or

- At least **$progressInterval** seconds has elapsed.

  Or

- The **$progressContent** mask has a bit set which matches a bit set in the message class provided with this message.

The **$progressCallOut** function is executed during **mmbackup** with a single argument consisting of the following colon-separated values:

```
"$JOB:$FS:$SERVER:$NODENAME:$PHASE:$BCKFILES:$CHGFILES:$EXPFILES:\
 $FILESBACKEDUP:$FILESEXPIRED:$ERRORS:$TIME"
```

Where:

JOB
Specifies the literal backup string to identify this component.

FS Specifies the file system device name.

SERVER
Specifies the IBM Spectrum Protect server currently used for backup.

NODENAME
Specifies the name of the node where **mmbackup** was started.

PHASE
Specifies either **synchronizing**, **scanning**, **selecting files**, **expiring**, **backing up**, **analyzing**, or **finishing**.

BCKFILES
Specifies the total number of files already backed up, or stored, on the IBM Spectrum Protect server. Starts as the count of all normal mode records in all the current shadow databases in use. If **QUERY** is being executed, it will start as the count of files found on the IBM Spectrum Protect server. It will stay constant until the backup job is complete.

CHGFILES
Specifies the number of changed files. This value starts as 0 and changes to the total number of changed files destined for the current server, and then stays at that value.

EXPFILES
Specifies the number of expired files. This value starts as 0 and changes to the total number of files marked for expiration at the current server, and then stays at that value.

FILESBACKEDUP
Specifies the number of files that were backed up during this backup job. This value remains 0 until phase **backing up** is reached, and then it increases until **dsmc** finishes. This value increases while **dsmc selective** jobs are running, and is calculated by IBM Spectrum Protect output. If the backup job fails before completion, some output may indicate files backed up but not counted. This value always increases.

FILESEXPIRED
Specifies the number of files that expired during this **expire** job. This value remains 0 until phase **expiring** is reached, and then it increases until **dsmc** finishes. This value increases while **dsmc** expire jobs are running, and is calculated by IBM Spectrum Protect output. If the backup job fails before completion, some output may indicate files expired but not counted. This value always increases.

ERRORS
Specifies the number of errors, not warnings or informational messages, that occurred during processing.

TIME
Specifies the time stamp as a **ctime** or number of seconds since the Epoch.

Backing up a file system using the GPFS policy engine

If IBM Spectrum Protect is not available, you can use the fast scan capabilities of the GPFS policy engine to generate lists of files to be backed up and provide them as input to some other external storage manager.

This process typically includes:
- Creating a policy file with LIST rules and associated criteria to generate the desired lists
- Optionally, creating a snapshot to obtain a consistent copy of the file system at a given point in time
- Running the **mmapplypolicy** command to generate the lists of files to back up
- Invoking the external storage manager to perform the actual backup operation

For more information on GPFS policies and rules refer to Chapter 25, "Information lifecycle management for IBM Spectrum Scale," on page 355.

Backing up file system configuration information

The **mmbackupconfig** command can be used to back up vital file system configuration information. This information can later be used to restore the layout and major characteristics of the file system.

The **mmbackupconfig** command creates a file that includes:
- Disk information (NSD names, sizes, failure groups)
- Storage pool layout
- Filesets and junction points
- Policy file rules
- Quota settings and current limits
- File system parameters (block size, replication factors, number of inodes, default mount point, and so on)

The output file generated by the **mmbackupconfig** command is used as input to the **mmrestoreconfig** command.

Note: The **mmbackupconfig** command only backs up the file system configuration information. It does not back up any user data or individual file attributes.

It is recommended that you store the output file generated by **mmbackupconfig** in a safe location.

Using APIs to develop backup applications

You can develop backup applications using APIs

IBM has supplied a set of subroutines that are useful to create backups or collect information about all files in a file system. Each subroutine is described in *Programming interfaces* in *IBM Spectrum Scale: Command and Programming Reference*. These subroutines are more efficient for traversing a file system, and provide more features than the standard POSIX interfaces. These subroutines operate on a global snapshot or on the active file system. They have the ability to return all files, or only files that have changed since some earlier snapshot, which is useful for incremental backup.

A typical use of these subroutines is the following scenario:

1. Create a global snapshot using the `mmcrsnapshot` command. For more information on snapshots, see the *IBM Spectrum Scale: Command and Programming Reference*.
2. Open an inode scan on the global snapshot using the `gpfs_open_inodescan()` or `gpfs_open_inodescan64()` subroutine.
3. Retrieve inodes using the `gpfs_next_inode()` or `gpfs_next_inode64()` subroutine.
4. Read the file data:
   a. Open the file using the `gpfs_iopen()` or `gpfs_iopen64()` subroutine.
   b. Read the file using the `gpfs_iread()`, `gpfs_ireadx()`, `gpfs_ireaddir()`, or `gpfs_ireaddir64()` subroutines.
   c. Close the file using the `gpfs_iclose()` subroutine.

The `gpfs_ireadx()` subroutine is more efficient than `read()` or `gpfs_iread()` for sparse files and for incremental backups. The `gpfs_ireaddir()` or `gpfs_ireaddir64()` subroutine is more efficient than `readdir()`, because it returns file type information. There are also subroutines for reading symbolic links, `gpfs_ireadlink()` or `gpfs_ireadlink64()` and for accessing file attributes, `gpfs_igetattrs()`.

Scale Out Backup and Restore (SOBAR)

Scale Out Backup and Restore (SOBAR) is a specialized mechanism for data protection against disaster only for GPFS file systems that are managed by IBM Spectrum Protect Hierarchical Storage Management (HSM). For such systems, the opportunity exists to *pre-migrate* all file data into the HSM storage and take a snapshot of the file system structural metadata, and save a backup image of the file system structure. This metadata image backup, consisting of several image files, can be safely stored in the backup pool of the IBM Spectrum Protect server and later used to restore the file system in the event of a disaster.

The SOBAR utilities include the commands **mmbackupconfig**, **mmrestoreconfig**, **mmimgbackup**, and **mmimgrestore**. The **mmbackupconfig** command will record all the configuration information about the file system to be protected and the **mmimgbackup** command performs a backup of GPFS file system metadata. The resulting configuration data file and the metadata image files can then be copied to the IBM Spectrum Protect server for protection. In the event of a disaster, the file system can be recovered by recreating the necessary NSD disks, restoring the file system configuration with the **mmrestoreconfig** command, and then restoring the image of the file system with the **mmimgrestore** command. The **mmrestoreconfig** command must be run prior to running the **mmimgrestore** command. SOBAR will reduce the time needed for a complete restore by utilizing all available bandwidth and all available nodes in the GPFS cluster to process the image data in a highly parallel fashion. It will also permit users to access the file system before all file data has been restored, thereby minimizing the file system down time. Recall from HSM of needed file data is performed automatically when a file is first accessed.

These commands cannot be run from a Windows node.

For the full details of the SOBAR procedures and requirements, see *Scale Out Backup and Restore (SOBAR)* in *IBM Spectrum Scale: Command and Programming Reference*.

Scheduling backups using IBM Spectrum Protect scheduler

The IBM Spectrum Protect scheduler typically utilizes the IBM Spectrum Protect Backup-Archive client backup commands that should be avoided in the IBM Spectrum Scale setup. Instead, you can configure a IBM Spectrum Protect client schedule to call a script as described in the following steps.

For scheduled events to occur on the client, you must configure the client scheduler to communicate with the IBM Spectrum Protect server. This is in addition to the following steps. For example, you might need to start the dsmcad service or add `MANAGEDSERVICES schedule` to the corresponding IBM Spectrum Protect

stanza in `dsm.sys` on the client node. For more information, see *Configuring the scheduler* in the IBM Spectrum Protect documentation on IBM Knowledge Center.

For the following steps, these example values are assumed:

```
client-node-proxyname (asnodename)          => proxy-cluster1
Node to be used for the schedule (aka nodename) => gpfs-node1
tsm server name                             => tsm1
file system to be backed up                 => gpfs0
global snapshot name (created for backup job) => BKUPsnap
schedule name on the TSM server             => proxy-cluster1_sched
```

1. On the IBM Spectrum Protect server, define the schedule using the following command.

   ```
   define schedule standard proxy-cluster1_sched type=client action=command objects=/usr/bin/
   my-mmbackup-script.sh starttime=05:00:00 startdate=today
   ```

2. On the IBM Spectrum Protect server, associate the schedule with the IBM Spectrum Scale proxy node using the following command.

   ```
   define association standard proxy-cluster1_sched proxy-cluster1
   ```

3. Create the backup script on the IBM Spectrum Scale node.

   Note: The following example script must be extended to log the output into files so that verification or troubleshooting can be done afterwards. Additional options such as `--noquote` might be needed depending on the specific needs of the environment.

   ```
   #!/bin/bash
   /usr/lpp/mmfs/bin/mmcrsnapshot gpfs0 BKUPsnap
   /usr/lpp/mmfs/bin/mmbackup gpfs0 —t incremental —-tsm-servers tsm1
   /usr/lpp/mmfs/bin/mmdelsnapshot gpfs0 BKUPsnap
   ```

4. On one of the IBM Spectrum Protect client nodes, verify the schedule using the following command.

   ```
   dsmc q sched
   ```

Configuration reference for using IBM Spectrum Protect with IBM Spectrum Scale

When using the IBM Spectrum Protect client in an IBM Spectrum Scale environment, several options in the `dsm.sys` and `dsm.opt` configuration files need to be taken into consideration.

Note: Refer to the latest IBM Spectrum Protect documentation on IBM Knowledge Center for the latest information on the mentioned settings.

Options in the IBM Spectrum Protect configuration file dsm.sys

This topic describes the options in the IBM Spectrum Protect configuration file dsm.sys.

Important: While the IBM Spectrum Protect client configuration file `dsm.sys` can contain node specific information, it cannot simply be copied from node to node without touching or correcting the corresponding node specific information.

Exclude or include options

File path name patterns that do not need to be backed up might be excluded by corresponding exclude statements. For example, temporary files. While IBM Spectrum Protect provides options for excluding and including, the usage of include options must be avoided when **mmbackup** is used. The reason is that **mmbackup** processing works properly with exclude statements but misinterpretations can arise when both, include and exclude, options are used together and in worst case have overlapping pattern sequences.

Note: Defining a large number of exclude rules can negatively impact the performance of backup.

Do not add exclude statements for snapshots as snapshots are specially handled automatically by **mmbackup** and IBM Spectrum Protect options when needed.

mmbackup excludes the following folders from the scan by default and these need not be explicitly excluded in the `dsm.sys` file or on the IBM Spectrum Protect server:

- `.mmbackup*` - folder in location specified by **MMBACKUP_RECORD_ROOT** such as `/ibm/gpfs0/.mmbackupCfg`
- `.mmLockDir` - folder in the root of the file system
- `.SpaceMan` - folder anywhere in the file system
- `.TsmCacheDir` - folder anywhere in the file system

Special consideration is needed when IBM Spectrum Protect server management class definitions are used. The corresponding include statements must be applied to any `dsm.sys` and not applied on the IBM Spectrum Protect server.

IBM Spectrum Protect users might be familiar with dynamic management class assignments available when using IBM Spectrum Protect **dsmc** commands to backup files. This is not the case with **mmbackup**. Only objects identified by **mmbackup** as requiring a backup will get the needed management class update that results when the administrator alters the management class assignment in the `dsm.sys` file. Therefore, only by running a complete backup of all affected objects can a management class update be guaranteed.

Despite the recommendation to never utilize the include statements in `dsm.sys`, when a IBM Spectrum Protect management class designation is needed, the use of an include statement with the management class specification is required. In these cases, do the following steps:

1. In the IBM Spectrum Protect client configuration file `dsm.sys`, arrange the include and exclude statements as follows:

   a. Place all the include statement first in the file along with the management class definitions.

   b. Add the exclude statements below the include statements.

   c. Ignore the ordering precedence rules defined in the IBM Spectrum Protect documentation regarding the ordering of these statements. Management class include statements must be listed above the exclude statements to work properly with **mmbackup**.

   Note: Do not add include statements after exclude statements. Do not add exclude statements before include statements.

2. Before starting the **mmbackup** job, set the following environment variable:

   ```
   export MMBACKUP_IGNORE_INCLUDE=1
   ```

Note:

- The include statements have no effect on the file system scan candidate selection in **mmapplypolicy** because the rules for include do not result in SQL statements being generated with **MMBACKUP_IGNORE_INCLUDE** activated.
- The include statements do not overrule the exclude statements which can be the case sometimes with **mmapplypolicy** policy rules generated from include and exclude formulation in IBM Spectrum Protect. It is recommended to never have overlapping patterns of any type with both include and exclude statements.

Usage of a IBM Spectrum Protect proxy node (asnodename option)

In a cluster, an operation that needs to scale is usually executed on more than one node, for example backup activities. To utilize the services of a IBM Spectrum Protect server from any of the configured cluster backup nodes, the administrator needs to specify a proxy node. This proxy node needs to be created on the IBM Spectrum Protect server similar to all other cluster backup nodes that need to be registered on the IBM Spectrum Protect server before they can be used. On all cluster backup nodes, set

the `asnodename` option for the desired proxy-client node to be used in the corresponding stanza of the `dsm.sys` configuration file.

Important IBM Spectrum Protect client configuration option

Option name	Remarks	Context
ASNODENAME $client-node-proxyname	Use the proxy node name (asnodename) instead of the cluster node name (nodename) to process cluster operations independent of a node name that is required for restore processing.	General

Options in the IBM Spectrum Protect configuration file dsm.opt

This topic describes the options in the IBM Spectrum Protect configuration file dsm.opt.

Special character handling

For IBM Spectrum Scale file systems with special characters frequently used in the names of files or directories, backup failures might occur. Known special characters that require special handling include: *, ?, ", ', carriage return, and the new line character.

In such cases, enable the IBM Spectrum Protect client options `WILDCARDSARELITERAL` and `QUOTESARELITERAL` on all nodes that are used in backup activities and make sure that the **mmbackup** option `--noquote` is used when invoking **mmbackup**.

Note: The characters control-X and control-Y are not supported by IBM Spectrum Protect. Therefore, the use of these characters in file names in IBM Spectrum Scale file systems results in these files not getting backed up to IBM Spectrum Protect.

Important IBM Spectrum Protect client configuration options

Option name	Remarks	Context
QUOTESARELITERAL [YES │ NO]	Requires the use of **mmbackup** with option `--noquote` if this is set to YES.	General
WILDCARDSARELITERAL [YES │ NO]	To handle the wildcard characters * and ? in file and folder names.	General
HSMDISABLEAUTOMIGDAEMONS [YES │ NO]	To prevent the IBM Spectrum Protect for Space Management **automigration** daemons from starting. Instead, the **mmapplypolicy** scan engine is used to identify migration candidates.	IBM Spectrum Protect for Space Management

Option name	Remarks	Context
SKIPACLUPDATECHECK [YES │ NO]	Requires UPDATECTIME to be enabled if this is set to YES. Using the SKIPACLUPDATECHECK option also omits checking for changes in the extended attributes (EAs) on Linux and AIX systems. Using this setting ensures that a file only gets backed up when the content of the file changes, not when only the ACL or EAs change. The backup of file after content changes then also includes the current ACL or EAs of the file.	General
SKIPACL [YES │ NO]	Requires UPDATECTIME to be enabled if this is set to YES. Using the skipacl option also omits EAs on Linux and AIX systems. Using this option can be considered when static ACL structures are used that can be reestablished through another tool or operation external to the IBM Spectrum Protect restore operation. If you are using this approach, ensure that the ACL is restored or established by inheritance, to avoid an unauthorized access to a recently restored file or directory. After enabling this option the ACL or EA is no longer backed up.	General
UPDATECTIME [YES │ NO]	This is to check the change time (ctime) attribute during a backup or archive operation. It is required to perform operations such as determining ACL changes.	General

Base IBM Spectrum Protect client configuration files for IBM Spectrum Scale usage

This topic lists all the Base IBM Spectrum Protect client configuration files and their examples for IBM Spectrum Scale.

Important: While the IBM Spectrum Protect client configuration file dsm.sys can contain node specific information, it cannot simply be copied from node to node without touching or correcting the corresponding node specific information.

The following are example contents of IBM Spectrum Protect configuration files.

Contents of dsm.sys

Note: Substitute the variables starting with '$' with your own required value. See the following example values of variables.

```
Servername $servername
    COMMMethod        TCPip
    TCPPort           $serverport
    TCPServeraddress  $serverip
*   TCPAdminport      $serveradminport
    TCPBuffsize       512
    PASSWORDACCESS    generate
* Place your exclude rules here or configure as cloptset on TSM server
    ERRORLOGName      $errorlog
    ASNODENAME        $client-node-proxyname
    NODENAME          $localnodename
```

Example values of variables used in dsm.sys

```
serverport=1500
serverip=myTSMserver.mydomain.org        OR     serverip=1.2.3.4
serveradminport=1526
errorlog=/var/log/mylogs/dsmerror.log
client-node-proxyname=proxy-cluster1
localnodename=gpfs-node1
```

Contents of dsm.opt

```
* Special character test flags
QUOTESARELITERAL YES
WILDCARDSARELITERAL YES
* to take traces just remove the * from the next two lines:
*TRACEFLAG SERVICE
*TRACEFILE /tmp/tsmtrace.txt
```

Contents of dsm.opt when IBM Spectrum Protect for Space Management is used

```
* HSM: Write extObjID to DMAPI attribute 'IBMexID' for migrated/pre-migrated files
HSMEXTOBJIDATTR yes
* HSM: Deactivate HSM Automigration and Scout search engine as this will be done by GPFS
HSMDISABLEAUTOMIGDAEMONS YES
* HSM file aggregation of small files
HSMGROUPedmigrate yes
* HSM: Determines if files that are less than 2 minutes old can be migrated during selective migration
hsmenableimmediatemigrate yes
```

Restoring a subset of files or directories from a local file system snapshot

You can restore a subset of files or directories from a local snapshot of a file system in case of accidental deletion.

Ensure the following before you begin:
- You have the full path to the files or directories that you want to restore. The path must include the file system to which these files or directories belong.
- You know which snapshot contains the files or directories that you want to restore.
- You have created a restore directory to which these files or directories are to be restored to avoid accidentally overwriting files or directories.

For information on how to create and maintain snapshots, see Chapter 26, "Creating and maintaining snapshots of file systems," on page 413

Use these steps to restore files or directories from a local file system snapshot.
1. Use the `mmlssnapshot` *device* command to list the snapshots in the file system and make a note of the snapshot that contains the files and directories that you want to restore.

   device is the name of the file system.

```
# mmlssnapshot fs1

Snapshots in file system fs1:
Directory        SnapId  Status   Created                     Fileset
fileset_test1    1       Valid    Mon Mar 23 09:20:37 2015    nfs-ganesha
filesystem_test2 2       Valid    Mon Mar 23 11:12:59 2015
```

2. Use the **mmsnapdir** *device* command to obtain the name of the snapshot directory for the file system snapshot that you have identified.

   In the following example, the fileset snapshot directory is called `.snapshots`.

   ```
   # mmsnapdir fs1

   Fileset snapshot directory for "fs1" is ".snapshots" (root directory only)
   Global snapshot directory for "fs1" is ".snapshots" in root fileset
   ```

3. Use the **mmlsfs** *device* -T command to determine the default mount point of the file system.

   In the following example, the default mount point is /gpfs/fs1.

   ```
   # mmlsfs fs1 -T

   flag  value      description
   ----  ------     ---------------
   -T    /gpfs/fs1  Default mount point
   ```

4. Use the full path to the files and directories that you want to restore and the default mount point that you have determined to obtain the truncated path to the files and directories.

   For example:
   ```
   Full path to the file:  /gpfs/fs1/nfs-ganesha/test1/
   Default mount point:    /gpfs/fs1
   Truncated path:         /nfs-ganesha/test1/
   ```

5. Change the directory to the full snapshot path of the file or the directory to verify.

   The full snapshot path is:

   filesystem_default_mountpoint/snapshot_directory/snapshot_name/truncated_path

   The full snapshot path using examples in the preceding steps is:

   ```
   /gpfs/fs1/.snapshots/filesystem_test2/nfs-ganesha/test1/
   ```

6. Do one of the following steps depending on whether you want to restore a file or a directory:
   - If you want to restore a file, use the following command:

     cp -p *full_snapshot_path/file_name restore_directory*
   - If you want to restore a directory, change the directory to the *restore_directory* and use the following command:

     tar -zcf *tar_file_name full_snapshot_path/directory_name*

Restoring a subset of files or directories from a local fileset snapshot

You can restore a subset of files or directories from a local snapshot of an independent fileset in case of accidental deletion.

Ensure the following before you begin:
- You have the full path to the files or directories that you want to restore. The path must include the file system to which these files or directories belong.
- You know which snapshot contains the files or directories that you want to restore.
- You have created a restore directory to which these files or directories are to be restored to avoid accidentally overwriting files or directories.

For information on how to create and maintain snapshots, see Chapter 26, "Creating and maintaining snapshots of file systems," on page 413

Use these steps to restore files or directories from a local fileset snapshot.

1. Use the **mmlssnapshot** *device* command to list the snapshots in the file system and make a note of the snapshot that contains the files and directories that you want to restore.

   device is the name of the file system.

   ```
   # mmlssnapshot fs1

   Snapshots in file system fs1:
   Directory       SnapId  Status   Created                     Fileset
   fileset_test1   1       Valid    Mon Mar 23 09:20:37 2015    nfs-ganesha
   filesystem_test2 2      Valid    Mon Mar 23 11:12:59 2015
   ```

2. Use the **mmsnapdir** *device* command to obtain the name of the snapshot directory for the fileset snapshot that you have identified.

   In the following example, the fileset snapshot directory is called .snapshots.

   ```
   # mmsnapdir fs1

   Fileset snapshot directory for "fs1" is ".snapshots" (root directory only)
   Global snapshot directory for "fs1" is ".snapshots" in root fileset
   ```

3. Use the **mmlsfileset** *device* command to verify that the fileset status is linked and to determine the full path of the fileset.

   In the following example, all filesets are linked and the paths are in the third column.

   ```
   # mmlsfileset fs1

   Filesets in file system 'fs1':
   Name          Status   Path
   root          Linked   /gpfs/fs1
   nfs-ganesha   Linked   /gpfs/fs1/nfs-ganesha
   nfs-ganesha2  Linked   /gpfs/fs1/nfs-ganesha2
   nfs-ganesha3  Linked   /gpfs/fs1/nfs-ganesha3
   nfs-ganesha4  Linked   /gpfs/fs1/nfs-ganesha4
   ```

4. Use the full path to the files and directories that you want to restore and the fileset path that you have determined to obtain the truncated path to the files and directories.

   For example:

   ```
   Full path to the file:  /gpfs/fs1/nfs-ganesha/test1/
   Fileset path:           /gpfs/fs1/nfs-ganesha
   Truncated path:         /test1/
   ```

5. Change the directory to the full snapshot path of the file or the directory to verify.

   The full snapshot path is:

   fileset_path/*snapshot_directory*/*snapshot_name*/*truncated_path*

   The full snapshot path using examples in the preceding steps is:

   /gpfs/fs1/nfs-ganesha/.snapshots/fileset_test1/test1/

6. Do one of the following steps depending on whether you want to restore a file or a directory:

   - If you want to restore a file, use the following command:

     cp -p *full_snapshot_path*/*file_name restore_directory*

   - If you want to restore a directory, change the directory to the *restore_directory* and use the following command:

     tar -zcf *tar_file_name full_snapshot_path*/*directory_name*

Restoring a subset of files or directories from local snapshots using the sample script

You can restore a subset of files or directories from local snapshots using a sample script in case of accidental deletion.

- The **mmcdpsnapqueryrecover** sample script only works on the Linux operating system.

- The sample script retrieves files or directories from all file system and fileset snapshots on the system and presents a list of files that you can choose to restore.
- Regular files are simply copied into the user-specified directory. If the user specifies a directory to be retrieved, the directory is copied into the user-specified directory as a compressed tar file.
- Files and directories that contain spaces in their names can also be retrieved.

Use the **mmcdpsnapqueryrecover** sample script to restore files or directories from snapshots into the user-specified `restorePath` directory as follows.

1. Use the following command to list all copies of a file or directory in a file system or fileset snapshot.

    ```
    /usr/lpp/mmfs/samples/ilm/mmcdpsnapqueryrecover.sh Device \
    --file-path fsPath --destination-dir restorePath
    ```

    Where:
    - *device* is the name of the file system.
    - *file-path* is the full file path.
    - *destination-dir* is the full path of the restore directory.

    For example, to get all copies of the file `/gpfs0/gplssnapshot` in the file system `gpfs0` and with `/opt` as the restore directory, enter the following:

    ```
    /usr/lpp/mmfs/samples/ilm/mmcdpsnapqueryrecover.sh /dev/gpfs0 \
    --file-path /gpfs0/gplssnapshot --destination-dir /opt
    ```

    All copies of the specified file are listed as follows:

    ```
    Found regular file in filesystem snapshot: restorFiles1
    1) 5743 Jan 9 08:34 /gpfs0/.snapshots/restorFiles1/gplssnapshot

    Found regular file in filesystem snapshot: restorFiles3
    2) 5882 Jan 9 08:34 /gpfs0/.snapshots/restorFiles3/gplssnapshot

    Found regular file in filesystem snapshot: Restore1
    3) 5886 Jan 14 12:33 /gpfs0/.snapshots/Restore1/gplssnapshot

    Found regular file in filesystem snapshot: Restore2
    4) 5886 Jan 14 12:33 /gpfs0/.snapshots/Restore2/gplssnapshot

    Found regular file in filesystem snapshot: global1
    5) 5886 Jan 14 12:33 /gpfs0/.snapshots/global1/gplssnapshot

    Which copy of the file/directory (1-5) would you like to restore?
    ```

2. From the list, select the file that you want to restore by entering the corresponding number. For example:

    ```
    Which copy of the file/directory (1-5) would you like to restore? 2
    ```

    The copy number 2 is restored to the `/opt` directory.

Creating and managing file systems using GUI

You can create, view, and modify file systems.

A file system consists of a set of disks that are used to store file metadata as well as data and structures, including quota files and recovery logs. The file system achieves high performance I/O in the following ways:

- Stripes data across multiple disks that are attached to multiple nodes:
  - All data in the file system is read and written in wide parallel stripes.
  - The block size for the file system determines the size of the block writes.
  - The block size that is specified at the time of file system creation, is global across the file system, and cannot be changed after the file system is defined

- Optimizes for small block write operations; a block is also subdivided into subblocks, so that multiple small block application writes can be aggregated and stored in a file system block, without wasting space in the block.
- Provides a high performance metadata (inode) scan engine to scan the file system rapidly to enable fast identification of data that needs to be managed or migrated in the automated tiered storage environment.
- Supports a large block size that can be configured by the administrator to fit I/O requirements:
  - Typical block sizes are the default (4 MiB), which is suitable for most workloads, especially mixed small random and large sequential workloads.
  - For large sequential workloads, the file system can optionally be defined with block sizes up to 16 MiB.
- Uses advanced algorithms that improve read-ahead and write-behind file functions for caching.
- Uses block-level locking based on a sophisticated token-management system that provides data consistency, while allowing multiple application nodes concurrent access to the files.

Creating and deleting file system

Use the **Create File System** option available in the **Files** > **File Systems** page to create file systems.

Deleting a file system removes all of the data on that file system. Use caution when performing this task. To delete a file system, select the file system to be deleted and then select **Delete** from the **Actions** menu.

File system monitoring options

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 IBM Spectrum Scale 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 below:
   - 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 health status, performance details, and other important configuration aspects of file systems available in the system. The following important details are available in the file system table:
   - File systems configured in the system
   - Health status. The detailed information about the events reported against each file system are available in the **Events** tab of the file system detailed view.
   - Capacity information
   - Certain information of remote file systems that are mounted from a remote cluster.
   - Mount status, mount configuration, and number of local and remote mounts. .

- Number of pools that are part of the file system. A file system consists of one or more pools. Detailed information of pools of a file system are available in the **Pools** tab of the file system detailed view.
- Number of NSDs that are part of the file system. Detailed information of NSDs of a file system are available in the **NSDs** tab of the file system detailed view.
- Performance data. 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 above the table allows to refresh the table with more recent data. The detailed performance details per node, pool, and NSDs are available in the detailed view of the file system.
- Protocols that are used to export or share the data stored in the file system.
- Number of nodes on which the file system is mounted. Details specific to each node on which the file system is mounted are available in the detailed view of the file system. You can also mount or unmount the file system from the detailed view.

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.
- **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 IBM Spectrum Scale object storage on the file system.
- **Properties:** Provides details of the file system attributes. You can also utilize the **Automatic mount** option to configure the automatic mount mode of the file system.

Managing access control

You can control the access to files and directories in a file system by defining access control lists (ACLs). ACLs can be inherited within a file system. The link path of the file system does not inherit any ACL from a parent path. Therefore, you can set the ACL of the file system link path using the **Edit Access Control** option.

When creating a file system, a default ACL is set. To modify the access controls defined for a file system, right-click the file system that is listed in the file system view and select **Edit Access Control**. The owner, owning group, and access control list cannot be modified if the directory is not empty. Users with the role *Dataaccess* are allowed to modify owner, group, and ACL even when the directory is not empty.

Mounting or unmounting a file system

You can use the IBM Spectrum Scale GUI to mount or unmount individual file systems or multiple file systems on the selected nodes. Use the **Files** > **File Systems**, **Files** > **File Systems** > **View Details** > **Nodes**, or **Nodes** > **View Details** > **File Systems** page in the GUI to mount or unmount a file system.

The GUI has the following options related to mounting the file system:

1. Mount local file systems on nodes of the local IBM Spectrum Scale cluster.
2. Mount remote file systems on local nodes.
3. Select individual nodes, protocol nodes, or nodes by node class while selecting nodes on which the file system needs to be mounted.
4. Prevent or allow file systems from mounting on individual nodes.

   Do the following to prevent file systems from mounting on a node:

   a. Go to **Nodes** .
   b. Select the node on which you need to prevent or allow file system mounts.
   c. Select **Prevent Mounts** from the **Actions** menu.
   d. Select the required option and click **Prevent Mount** or **Allow Mount** based on the selection.
5. Configure automatic mount option. The automatic configure option determines whether to automatically mount file system on nodes when GPFS daemon starts or when the file system is accessed for the first time. You can also specify whether to exclude individual nodes while enabling the automatic mount option. To enable automatic mount, do the following:

   a. From the **Files** > **File Systems** page, select the file system for which you need to enable automatic mount.
   b. Select **Configure Automatic Mount** option from the **Actions** menu.
   c. Select the required option from the list of automatic mount modes.
   d. Click **Configure**.

   Note: You can configure automatic mount option for a file system only if the file system is unmounted from all nodes. That is, you need to stop I/O on this file system to configure this option. However, you can include or exclude the individual nodes for automatic mount without unmounting the file system from all nodes.

You can utilize the following unmount features that are supported in the GUI:

1. Unmount local file system from local nodes and remote nodes.
2. Unmount a remote file system from the local nodes. When a local file system is unmounted from the remote nodes, the remote nodes can no longer be seen in the GUI. The **Files** > **File Systems** > **View Details** > **Remote Nodes** page lists the remote nodes that currently mount the selected file system. The selected file system can be a local or a remote file system but the GUI permits to umount only local file systems from the remote nodes.
3. Select individual nodes, protocol nodes, or nodes by node class while selecting nodes from which the file system needs to be unmounted.
4. Specify whether to force unmount. Selecting the **Force unmount option** while unmounting the file system unmounts the file system even if it is still busy in performing the I/O operations. Forcing the unmount operation affects the outstanding operations and causes data integrity issues. The IBM Spectrum Scale system relies on the native unmount command to carry out the unmount operation. The semantics of forced unmount are platform-specific. On certain platforms such as Linux, even when forced unmount is requested, file system cannot be unmounted if it is still referenced by system kernel. To unmount a file system in such cases, identify and stop the processes that are referencing the file system. You can use system utilities like *lsof* and *fuser* for this.

Some administrative actions like repairing file system structures by using the `mmfsck` command, require that the file system is unmounted on all nodes.

Policies

IBM Spectrum Scale provides a way to automate the management of files by using policies and rules. You can manage these policies and rules through the **Files** > **Information Lifecycle** page of the management GUI.

A policy rule is an SQL-like statement that tells the file system what to do with the data for a file in a specific pool if the file meets specific criteria. A rule can apply to any file being created or only to files being created within a specific fileset or group of filesets.

Chapter 14. File system format changes between versions of IBM Spectrum Scale

Every GPFS file system has a format number associated with it. This format number corresponds to the on-disk data structures of the file system and is an indicator of the supported file system functionality.

The file system format number is assigned when the file system is first created and is updated to the latest supported level after the file system is migrated with the **mmchfs -V** command.

The format number for a file system can be displayed with the **mmlsfs -V** command. If a file system was created with an older GPFS release, new functionality that requires different on-disk data structures is not enabled until you run the **mmchfs -V** command. Some new features might require you also to run the **mmmigrate** command.

Note: The **-V** parameter cannot be used to make file systems that were created before GPFS 3.2.1.5 available to Windows nodes. Windows nodes can mount only file systems that were created with GPFS 3.2.1.5 or later.

The **mmchfs -V** parameter requires the specification of one of two values, **full** or **compat**:

- Specifying **mmchfs -V full** enables all of the new functionality that requires different on-disk data structures. After this command, nodes in remote clusters that are running an older GPFS version will no longer be able to mount the file system.

  The **mmchfs -V full** command displays a warning as in the following example:

```
# mmchfs n03NsdOnFile36 -V full
You have requested that the file system be upgraded to
version 18.00 (5.0.0.0). This will enable new functionality but will
prevent you from using the file system with earlier releases of GPFS.
Do you want to continue?
```

- Specifying **mmchfs -V compat** enables only backward-compatible format changes. Nodes in remote clusters that were able to mount the file system before the format changes can continue to do so afterward.

In IBM Spectrum Scale 5.0.0.0, new file systems are created at format number 18.00. To update the format of an earlier file system to format number 18.00, issue the following command:

```
mmchfs Device -V full
```

where *Device* is the device name of the earlier file system. The following features of IBM Spectrum Scale 5.0.0.0 require a file system to be at format number 18.00 or later:

- Smaller subblock sizes for file systems that have a large data block size

  Note: This feature is supported only for file systems that are created at file system format number 18.00 or later. It is not supported for file systems that are updated to format number 18.00 or later from an earlier format number. For more information, see the parameter **-B** *BlockSize* in the topic *mmcrfs* in the *IBM Spectrum Scale: Command and Programming Reference*.

- File compression with the lz4 compression library
- File audit logging

In IBM Spectrum Scale v4.2.3.0, new file systems are created at format number 17.00. To update the format of an earlier file system to format number 17.00, issue the following command:

```
mmchfs Device -V full
```

| where *Device* is the device name of the earlier file system. The following features of IBM Spectrum Scale
| v4.2.3.0 require a file system to be at format number 17.00 or later:
| • Quality of Service for I/O (QoS)
| • File compression with zlib compression library
| • Information lifecycle management (ILM) for snapshots

If your current file system is at format number 14.20 (IBM Spectrum Scale 4.1.1), the set of enabled
features depends on the value specified with the **mmchfs -V** option:
• After running **mmchfs -V full**, the file system can support the following:
  – Enabling and disabling of quota management without unmounting the file system.
  – The use of fileset-level integrated archive manager (IAM) modes.
• There are no new features that can be enabled with **mmchfs -V compat**.

If your current file system is at format number 14.04 (GPFS 4.1.0.0), the set of enabled features depends
on the value specified with the **mmchfs -V** option:
• After running **mmchfs -V full**, the file system can support different block allocation map types on an
  individual storage-pool basis.
• There are no new features that can be enabled with **mmchfs -V compat**.

If your current file system is at format number 13.23 (GPFS 3.5.0.7), the set of enabled features depends
on the value specified with the **mmchfs -V** option:
• After running **mmchfs -V full**, the file system can support the following:
  – Directory block sizes can be up to 256 KB in size (previous maximum was 32 KB).
  – Directories can reduce their size when files are removed.
• There are no new features that can be enabled with **mmchfs -V compat**.

If your current file system is at format number 13.01 (GPFS 3.5.0.1), the set of enabled features depends
on the value specified with the **mmchfs -V** option:
• After running **mmchfs -V full**, the file system can support the following:
  – extended storage pool properties
  – File Placement Optimizer (FPO)
  – Storing small directories in the inode
  – Storing the data for small files in the inode
• There are no new features that can be enabled with **mmchfs -V compat**.

If your current file system is at format number 12.03 (GPFS 3.4), the set of enabled features depends on
the value specified with the **mmchfs -V** option:
• After running **mmchfs -V full**, the file system can support the following:
  – independent filesets and snapshots of individual independent filesets
  – active file management (AFM)
  – file clones (writable snapshots of a file)
  – policy language support for new attributes, variable names, and functions: OPTS clause for the SET
    POOL and RESTORE rules, encoding of path names via an ESCAPE clause for the EXTERNAL LIST
    and EXTERNAL POOL rules, GetEnv(), GetMMconfig(), SetXattr(), REGEX().
• There are no new features that can be enabled with **mmchfs -V compat**.

If your current file system is at format number 11.03 (GPFS 3.3), the set of enabled features depends on
the value specified with the **mmchfs -V** option:
• After running **mmchfs -V full**, the file system can support the following:

- – more than 2,147,483,648 files
- – fast extended attributes (which requires **mmmigratefs** to be run also)
- There are no new features that can be enabled with **mmchfs -V compat**.

If your current file system is at format number 10.00 (GPFS 3.2.0.0) or 10.01 (GPFS 3.2.1.5), after running **mmchfs -V**, the file system can support all of the features included with earlier levels, plus the following:
- new maximum number of filesets in a file system (10000)
- new maximum for the number of hard links per object (2**32)
- improved quota performance for systems with large number of users
- policy language support for new attributes, variable names, and functions: MODE, INODE, NLINK, RDEVICE_ID, DEVICE_ID, BLOCKSIZE, GENERATION, XATTR(), ATTR_INTEGER(), and XATTR_FLOAT()

If your current file system is at format number 9.03 (GPFS 3.1), after running **mmchfs -V**, the file system can support all of the features included with earlier levels, plus:
- fine grain directory locking
- LIMIT clause on placement policies

If your current file system is at format number 8.00 (GPFS 2.3), after running **mmchfs -V**, the file system can support all of the features included with earlier levels, plus:
- storage pools
- filesets
- fileset quotas

If your current file system is at format number 7.00 (GPFS 2.2), after running **mmchfs -V**, the file system can support all of the features included with earlier levels, plus:
- NFS V4 access control lists
- new format for the internal allocation summary files

If your current file system is at format number 6.00 (GPFS 2.1), after running **mmchfs -V**, the file system can support all of the features included with earlier levels, plus extended access control list entries (**-rwxc** access mode bits).

The functionality described in this topic is only a subset of the functional changes introduced with the different GPFS releases. Functional changes that do not require changing the on-disk data structures are not listed here. Such changes are either immediately available when the new level of code is installed, or require running the **mmchconfig release=LATEST** command. For a complete list, see the "Summary of changes" on page xxix.

Chapter 15. Managing disks

Use the following information to manage disks in IBM Spectrum Scale

Disks can have connectivity to each node in the cluster, be managed by network shared disk servers, or a combination of the two. For more information, see *mmcrnsd command* in the *IBM Spectrum Scale: Command and Programming Reference*. Also see, *Network Shared Disk (NSD) creation considerations* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Note: A LUN provided by a storage subsystem is a disk for the purposes of this documentation, even if the LUN is made up of multiple physical disks.

The disk related tasks performed on a GPFS file system include:
1. "Displaying disks in a GPFS cluster"
2. "Adding disks to a file system" on page 158
3. "Deleting disks from a file system" on page 158
4. "Replacing disks in a GPFS file system" on page 160
5. "Additional considerations for managing disks" on page 162
6. "Displaying GPFS disk states" on page 162
7. "Changing GPFS disk states and parameters" on page 163
8. "Changing your NSD configuration" on page 165
9. "Changing NSD server usage and failback" on page 166
10. "Enabling and disabling Persistent Reserve" on page 166

Displaying disks in a GPFS cluster

You can display the disks that belong to your GPFS cluster by issuing the **mmlsnsd** command.

The default is to display information for all disks defined to the cluster (**-a**). Otherwise, you may choose to display the information for a particular file system (**-f**) or for all disks which do not belong to any file system (**-F**).

To display the default information for all of the NSDs belonging to the cluster, enter:
```
mmlsnsd
```

The system displays information similar to:
```
File system  Disk name    NSD servers
---------------------------------------------------------------------
fs2          hd3n97       c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2          hd4n97       c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2          hd5n98       c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2          hd6n98       c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2          sdbnsd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2          sdcnsd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2          sddnsd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2          sdensd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2          sdgnsd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2          sdfnsd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2          sdhnsd       c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
(free disk) hd2n97        c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com
```

To find out the local device names for the disks, use the **mmlsnsd** command with the **-m** option. For example, issuing **mmlsnsd -m** produces output similar to this:

```
Disk name   NSD volume ID      Device        Node name                  Remarks
-------------------------------------------------------------------------------
hd2n97      0972846145C8E924   /dev/hdisk2   c5n97g.ppd.pok.ibm.com     server node
hd2n97      0972846145C8E924   /dev/hdisk2   c5n98g.ppd.pok.ibm.com     server node
hd3n97      0972846145C8E927   /dev/hdisk3   c5n97g.ppd.pok.ibm.com     server node
hd3n97      0972846145C8E927   /dev/hdisk3   c5n98g.ppd.pok.ibm.com     server node
hd4n97      0972846145C8E92A   /dev/hdisk4   c5n97g.ppd.pok.ibm.com     server node
hd4n97      0972846145C8E92A   /dev/hdisk4   c5n98g.ppd.pok.ibm.com     server node
hd5n98      0972846245EB501C   /dev/hdisk5   c5n97g.ppd.pok.ibm.com     server node
hd5n98      0972846245EB501C   /dev/hdisk5   c5n98g.ppd.pok.ibm.com     server node
hd6n98      0972846245DB3AD8   /dev/hdisk6   c5n97g.ppd.pok.ibm.com     server node
hd6n98      0972846245DB3AD8   /dev/hdisk6   c5n98g.ppd.pok.ibm.com     server node
```

Adding disks to a file system

Many file systems grow rapidly, so after creating a file system you might decide that more disk space is required.

Storage in a file system is divided in storage pools. The maximum size of any one disk that can be added to an existing storage pool is set approximately to the sum of the disk sizes when the storage pool is created. The actual value is shown in the **mmdf** command output.

Once a storage pool is created, the maximum size *cannot* be altered. However, you can create a new pool with larger disks, and then move data from the old pool to the new one.

When establishing a storage pool and when adding disks later to an existing storage pool, you should try to keep the sizes of the disks fairly uniform. GPFS allocates blocks round robin, and as the utilization level rises on all disks, the small ones will fill up first and all files created after that will be spread across fewer disks, which reduces the amount of prefetch that can be done for those files.

To add disks to a GPFS file system, first decide if you will:

1. Create new disks using the **mmcrnsd** command.

   In this case, you must also decide whether to create a new set of NSD and pools stanzas or use the rewritten NSD and pool stanzas that the **mmcrnsd** command produces. In a rewritten file, the disk usage, failure group, and storage pool values are the same as the values that are specified in the **mmcrnsd** command.

2. Select disks no longer in use in any file system. Issue the **mmlsnsd -F** command to display the available disks.

The disk may then be added to the file system using the stanza file as input to the **mmadddisk** command.

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

For more information, see the *mmadddisk command* and the *mmcrnsd command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Deleting disks from a file system

Before deleting a disk use the **mmdf** command to determine whether there is enough free space on the remaining disks to store the file system.

Note: See "Querying file system space" on page 129 for more information about diagnosing space problems.

Consider how fragmentation may increase your storage requirements, especially when the file system contains a large number of small files. A margin of 150 percent of the size of the disks being deleted should be sufficient to allow for fragmentation when small files predominate. For example, in order to delete a 400 GB disk from your file system, which contains user home directories with small files, you should first determine that the other disks in the file system contain a total of 600 GB of free space.

If you do not replicate your file system data, you should rebalance the file system using the **mmrestripefs -b** command. If you replicate your file system data, run the **mmrestripefs -r** command after the disk has been deleted. This ensures that all data will still exist with correct replication after the disk is deleted. The **mmdeldisk** command only migrates data that would otherwise be lost, not data that will be left in a single copy.

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

Do not delete stopped disks, if at all possible. Start any stopped disk before attempting to delete it from the file system. If the disk cannot be started you will have to consider it permanently damaged. You will need to delete the disk using the appropriate **mmdeldisk** options. If metadata was stored on the disk, you will need to execute **mmfsck** in offline mode afterwards.. For more information on handling this situation, see *NSD and underlying disk subsystem failures* in the *IBM Spectrum Scale: Problem Determination Guide*.

When deleting disks from a file system, the disks may or may not be available. If the disks being deleted are still available, GPFS moves all of the data from those disks to the disks remaining in the file system. However, if the disks being deleted are damaged, either partially or permanently, it is not possible to move all of the data and you will receive I/O errors during the deletion process. For instructions on how to handle damaged disks, see *Disk media failure* in the *IBM Spectrum Scale: Problem Determination Guide*.

Specify the file system and the names of one or more disks to delete with the **mmdeldisk** command. For example, to delete the disk **hd2n97** from the file system **fs2** enter:

```
mmdeldisk fs2 hd2n97
```

The system displays information similar to:

```
Deleting disks ...
Scanning system storage pool
Scanning file system metadata, phase 1 ...
19 % complete on Fri Mar 16 23:23:50 2012
100 % complete on Fri Mar 16 23:23:51 2012
Scan completed successfully.
Scanning file system metadata, phase 2 ...
46 % complete on Fri Mar 16 23:23:55 2012
93 % complete on Fri Mar 16 23:23:58 2012
100 % complete on Fri Mar 16 23:23:58 2012
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
19.50 % complete on Fri Mar 16 23:24:25 2012 ( 35777 inodes 1207 MB)
47.92 % complete on Fri Mar 16 23:24:49 2012 ( 199955 inodes 2966 MB)
50.05 % complete on Fri Mar 16 23:25:09 2012 ( 235356 inodes 3098 MB)
53.09 % complete on Fri Mar 16 23:25:31 2012 ( 261831 inodes 3286 MB)
55.12 % complete on Fri Mar 16 23:25:51 2012 ( 283815 inodes 3412 MB)
63.25 % complete on Fri Mar 16 23:26:12 2012 ( 319236 inodes 3915 MB)
```

```
63.27 % complete on Fri Mar 16 23:26:33 2012 ( 382031 inodes 6223 MB)
63.29 % complete on Fri Mar 16 23:27:03 2012 ( 699858 inodes 9739 MB)
100.00 % complete on Fri Mar 16 23:27:35 2012
Scan completed successfully.
Checking Allocation Map for storage pool 'system'
17 % complete on Fri Mar 16 23:27:42 2012
31 % complete on Fri Mar 16 23:27:47 2012
48 % complete on Fri Mar 16 23:27:52 2012
62 % complete on Fri Mar 16 23:27:57 2012
76 % complete on Fri Mar 16 23:28:02 2012
90 % complete on Fri Mar 16 23:28:07 2012
100 % complete on Fri Mar 16 23:28:08 2012
tsdeldisk completed.
mmdeldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

For syntax and usage information, refer to *mmdeldisk command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Replacing disks in a GPFS file system

Replacing an existing disk in a GPFS file system with a new one is the same as performing a delete disk operation followed by an add disk. However, this operation eliminates the need to restripe the file system following the separate delete disk and add disk operations as data is automatically moved to the new disk.

When replacing disks in a GPFS file system, first decide if you will:

1. Create new disks using the **mmcrnsd** command.

   In this case, you must also decide whether to create a new set of NSD and pools stanzas or use the rewritten NSD and pool stanzas that the `mmcrnsd` command produces. In a rewritten file, the disk usage, failure group, and storage pool values are the same as the values that are specified in the `mmcrnsd` command.

2. Select NSDs no longer in use by another GPFS file system. Issue the **mmlsnsd -F** command to display the available disks.

To replace a disk in the file system, use the **mmrpldisk** command. For example, to replace the NSD **hd3n97** in file system **fs2** with the existing NSD **hd2n97**, which is no longer in use by another file system, enter:

```
mmrpldisk fs2 hd3n97 hd2n97
```

The system displays information similar to:

```
Replacing hd3n97 ...

The following disks of fs2 will be formatted on node c33f2in01:
hd2n97: size 571398144 KB
Extending Allocation Map
Checking Allocation Map for storage pool 'system'
9 % complete on Fri Mar 16 23:33:29 2012
23 % complete on Fri Mar 16 23:33:34 2012
37 % complete on Fri Mar 16 23:33:40 2012
52 % complete on Fri Mar 16 23:33:45 2012
66 % complete on Fri Mar 16 23:33:50 2012
83 % complete on Fri Mar 16 23:33:55 2012
98 % complete on Fri Mar 16 23:34:00 2012
100 % complete on Fri Mar 16 23:34:00 2012
Completed adding disks to file system fs2.
Scanning system storage pool
Scanning file system metadata, phase 1 ...
13 % complete on Fri Mar 16 23:34:19 2012
100 % complete on Fri Mar 16 23:34:22 2012
Scan completed successfully.
```

```
Scanning file system metadata, phase 2 ...
29 % complete on Fri Mar 16 23:34:26 2012
67 % complete on Fri Mar 16 23:34:29 2012
100 % complete on Fri Mar 16 23:34:32 2012
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
8.21 % complete on Fri Mar 16 23:34:54 2012 ( 37741 inodes 770 MB)
22.65 % complete on Fri Mar 16 23:35:14 2012 ( 40182 inodes 2124 MB)
32.95 % complete on Fri Mar 16 23:35:34 2012 ( 160837 inodes 3090 MB)
35.15 % complete on Fri Mar 16 23:35:57 2012 ( 227991 inodes 3296 MB)
36.34 % complete on Fri Mar 16 23:36:17 2012 ( 265748 inodes 3408 MB)
37.34 % complete on Fri Mar 16 23:36:38 2012 ( 284398 inodes 3502 MB)
46.07 % complete on Fri Mar 16 23:37:04 2012 ( 310636 inodes 4320 MB)
61.41 % complete on Fri Mar 16 23:37:25 2012 ( 315141 inodes 5759 MB)
87.04 % complete on Fri Mar 16 23:37:50 2012 ( 350241 inodes 8163 MB)
87.06 % complete on Fri Mar 16 23:38:11 2012 ( 370562 inodes 10136 MB)
87.08 % complete on Fri Mar 16 23:38:42 2012 ( 392561 inodes 11982 MB)
87.10 % complete on Fri Mar 16 23:39:22 2012 ( 401049 inodes 13195 MB)
87.12 % complete on Fri Mar 16 23:40:14 2012 ( 1100590 inodes 15685 MB)
100.00 % complete on Fri Mar 16 23:40:57 2012
Scan completed successfully.
Checking Allocation Map for storage pool 'system'
10 % complete on Fri Mar 16 23:41:02 2012
26 % complete on Fri Mar 16 23:41:07 2012
33 % complete on Fri Mar 16 23:41:12 2012
44 % complete on Fri Mar 16 23:41:17 2012
68 % complete on Fri Mar 16 23:41:22 2012
100 % complete on Fri Mar 16 23:41:25 2012
Done
mmrpldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note: If you attempt to replace a stopped disk and the file system is not replicated, the attempt will fail.

However, you can replace a stopped disk if the file system is replicated. You can do so in one of the following ways:

- Deletion, addition, and rebalancing method:

    1. Use the **mmdeldisk** command to delete the stopped disk from the file system.

    2. Use the **mmadddisk** command to add a replacement disk.

    3. Use the **mmrestripefs -b** command to rebalance the file system.

    While this method requires rebalancing, it returns the system to a protected state faster (because it can use all of the remaining disks to create new replicas), thereby reducing the possibility of losing data.

    —Or—

- Direct replacement method:

    Use the **mmrpldisk** command to directly replace the stopped disk.

    The **mmrpldisk** command only runs at single disk speed because all data being moved must be written to the replacement disk. The data is vulnerable while the command is running, and should a second failure occur before the command completes, it is likely that some data will be lost.

For more information on handling this situation, see *Disk media failure* in the *IBM Spectrum Scale: Problem Determination Guide*.

Additional considerations for managing disks

If you delete, replace, or suspend a disk with strict replication enforced, you may receive an ENOSPC error when you create or append data to an existing file.

If you need to delete, replace, or suspend a disk and you need to write new data while the disk is offline, you can disable strict replication before you perform the disk action. However, data written while replication is disabled will not be properly replicated. Therefore, after you perform the disk action, you must re-enable strict replication and run the `mmrestripefs -r` command. To determine if a file system has strict replication enforced, issue the `mmlsfs -K` command.

Displaying GPFS disk states

You can display the current state of one or more disks in your file system by issuing the **mmlsdisk** command.

The information includes parameters that were specified on the **mmcrfs** command, and the current availability and status of the disks. For example, to display the current status of the disk **hd8vsdn100** in the file system **fs1**, enter:

```
mmlsdisk fs1 -d hd8vsdn100
```

Status is displayed in a format similar to:

```
disk          driver   sector failure holds   holds                             storage
name          type       size   group metadata data  status       availability pool
------------ -------- ------ ------- -------- ----- ------------ ------------ ------------
hd8vsdn100   nsd         512       1 no       yes   ready        up           sp1
```

For syntax and usage information, see the *mmlsdisk command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Disk availability

The following information lists the possible values of disk availability, and what they mean.

A disk's availability determines whether GPFS is able to read and write to the disk. There are four possible values for availability:

up The disk is available to GPFS for normal **read** and **write** operations.

down No **read** and **write** operations can be performed on the disk.

recovering
An intermediate state for disks coming up, during which GPFS verifies and corrects data. A **write** operation can be performed while a disk is in this state, but a **read** operations cannot, because data on the disk being recovered might be stale until the **mmchdisk start** command completes.

unrecovered
The disk was not successfully brought up.

Disk availability is automatically changed from **up** to **down** when GPFS detects repeated I/O errors. You can also change the availability of a disk by issuing the **mmchdisk** command.

Disk status

The following information lists the possible values for disk status, and what they mean.

Disk status controls data placement and migration. Status changes as a result of a pending delete operation, or when the **mmchdisk** command is issued to allow file rebalancing or re-replicating prior to disk replacement or deletion.

Disk status has seven possible values, but four are transitional:

ready   Normal status.

suspended
or
to be emptied
>   Indicates that data is to be migrated off this disk.

being emptied
>   Transitional status in effect while a disk deletion is pending.

emptied
>   Indicates that data is already migrated off this disk.

replacing
>   Transitional status in effect for old disk while replacement is pending.

replacement
>   Transitional status in effect for new disk while replacement is pending.

GPFS allocates space only on disks with a status of **ready** or **replacement**.

GPFS migrates data off disks with a status of **being emptied**, **replacing**, **to be emptied**, or **suspended** onto disks with a status of **ready** or **replacement**. During disk deletion or replacement, data is automatically migrated as part of the operation. Issue the **mmrestripefs** command to initiate data migration from a suspended disk.

See "Deleting disks from a file system" on page 158, "Replacing disks in a GPFS file system" on page 160, and "Restriping a GPFS file system" on page 128.

Changing GPFS disk states and parameters

You might find it necessary to change a disk's state if there is some indication of disk failure or if you need to restripe the file system.

Refer to "Displaying GPFS disk states" on page 162 for a detailed description of disk states. You can change both the availability and status of a disk using the **mmchdisk** command:
*   Change disk availability using the **mmchdisk** command and the **stop** and **start** options
*   Change disk status using the **mmchdisk** command and the **suspend** and **resume** options.

Issue the **mmchdisk** command with one of the following four options to change disk state:

resume
>   Informs GPFS that a disk previously suspended is now available for allocating new space. Resume a disk only when you suspended it and decided not to delete or replace it. If the disk is currently in a stopped state, it remains stopped until you specify the **start** option. Otherwise, normal **read** and **write** access to the disk resumes.

start
>   Informs GPFS that a disk previously stopped is now accessible. GPFS does this by first changing the disk availability from **down** to **recovering**. The file system metadata is then scanned and any missing updates (replicated data that was changed while the disk was down) are repaired. If this operation is successful, the availability is then changed to **up**.
>
>   If the metadata scan fails, availability is set to **unrecovered**. This could occur if other disks remain in **recovering** or an I/O error has occurred. Repair all disks and paths to disks. It is recommended that you run **mmfsck** at this point (For more information, see *mmfsck command* in the *IBM Spectrum Scale: Command and Programming Reference*). The metadata scan can then be re-initiated at a later time by issuing the **mmchdisk start** command again.

If more than one disk in the file system is down, they should all be started at the same time by using the **-a** option. If you start them separately and metadata is stored on any disk that remains down, the **mmchdisk start** command fails.

stop

Instructs GPFS to stop any attempts to access the specified disk. Use this option to inform GPFS that a disk has failed or is currently inaccessible because of maintenance. A disk's availability remains **down** until it is explicitly started with the **start** option.

suspend

or

empty

Instructs GPFS to stop allocating space on the specified disk. Place a disk in this state prior to disk deletion or replacement. This is a user-initiated state that GPFS will never use without an explicit command to change disk state.

Note: A disk remains suspended until it is explicitly resumed. Restarting GPFS or rebooting nodes does not restore normal access to a suspended disk.

The `empty` option is similar to the `suspend` option. For releases prior to GPFS 4.1.1, the output of the **mmlsdisk** command displays the status as `suspended`, as shown in the following example.

For example, to suspend the **hd8vsdn100** disk in the file system **fs1**, enter:

```
mmchdisk fs1 suspend -d hd8vsdn100
```

To confirm the change, enter:

```
mmlsdisk fs1 -d hd8vsdn100
```

The system displays information similar to:

```
disk          driver   sector failure holds    holds                              storage
name          type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
hd8vsdn100   nsd         512       7 yes      yes   suspended     up           system
```

From GPFS 4.1.1 onwards, the status in the **mmlsdisk**command is displayed as `to be emptied`, as shown in the following example:

For example, to set `to be emptied` state for **gpfs1nsd** disk of the file system **fs1**, enter:

```
mmchdisk fs1 empty -d gpfs1nsd
```

To confirm the change, enter:

```
mmlsdisk fs1 -d gpfs1nsd
```

The system displays information similar to:

```
disk      driver sector failure holds    holds                           storage
name      type     size   group metadata data  status        availability disk id pool     remarks
--------- ------ ------ ------ -------- ---- ------------- ------------ ------- -------- ---------
gpfs1nsd  nsd    512      -1   Yes      Yes  to be emptied up               1   system
gpfs2nsd  nsd    512      -1   Yes      Yes  to be emptied up               2   system   desc
```

You can also use the **mmchdisk** command with the **change** option to change the *Disk Usage* and *Failure Group* parameters for one or more disks in a GPFS file system. This can be useful in situations where, for example, a file system that contains only RAID disks is being upgraded to add conventional disks that are better suited to storing metadata. After adding the disks using the **mmadddisk** command, the metadata currently stored on the RAID disks would have to be moved to the new disks to achieve the desired performance improvement. To accomplish this, first the **mmchdisk change** command would be issued to change the *Disk Usage* parameter for the RAID disks to **dataOnly**. Then the **mmrestripefs** command would be used to restripe the metadata off the RAID device and onto the conventional disks.

For example, to specify that metadata should no longer be stored on disk **hd8vsdn100**, enter:
```
mmchdisk fs1 change -d "hd8vsdn100:::dataOnly"
```

To confirm the change, enter:
```
mmlsdisk fs1 -d hd8vsdn100
```

The system displays information similar to:
```
disk         driver   sector failure holds    holds                              storage
name         type       size   group metadata data  status        availability  pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
hd8vsdn100   nsd         512       1 no       yes   ready         up           sp1
```

For complete usage information, see the *mmchdisk command* and the *mmlsdisk command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Changing your NSD configuration

Use the following steps to change the NSD configuration.

Once your NSDs have been created, you may change the configuration attributes as your system requirements change. For more information about creating NSDs, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* and the *mmcrnsd command* in the *IBM Spectrum Scale: Command and Programming Reference*)

By issuing the **mmchnsd** command you can:
- Specify up to eight servers for an NSD that does not have one.
- Change the NSD server nodes specified in the server list.
- Delete the server list. The disk must now be SAN-attached to all nodes in the cluster on which the file system will be mounted.

You must follow these rules when changing NSDs:
- Identify the disks by the NSD names that were given to them by the **mmcrnsd** command.
- Explicitly specify values for all NSD servers in the list, even if you are only changing one of the values.
- Unmount the file system that contains the NSD being changed prior to issuing the **mmchnsd** command.
- Connect the NSD to the new nodes prior to issuing the **mmchnsd** command.
- **mmchnsd** cannot change the *DiskUsage* or *FailureGroup* for an NSD. Use the **mmchdisk** command to change these attributes.
- To move a disk from one storage pool to another, use the **mmdeldisk** and **mmadddisk** commands.
- You cannot change the name of the NSD.

For example, to assign node **k145n07** as an NSD server for disk **gpfs47nsd**:
1. Make sure that **k145n07** is not already assigned to the server list by issuing the **mmlsnsd** command.
   ```
   mmlsnsd -d "gpfs47nsd"
   ```

   The system displays information similar to:
   ```
    File system  Disk name    NSD server nodes
    ---------------------------------------------------------------
    fs1          gpfs47nsd    k145n09
   ```
2. Unmount the file system on all nodes and ensure that the disk is connected to the new node (**k145n07**).
3. Issue the **mmchnsd** command:

```
mmchnsd "gpfs47nsd:k145n09,k145n07"
```

4. Verify the changes by issuing the **mmlsnsd** command:

```
mmlsnsd -d gpfs47nsd
```

The system displays information similar to:

```
  File system   Disk name    NSD servers
---------------------------------------------------------------------------
  fs2           gpfs47nsd    k145n09.ppd.pok.ibm.com,k145n07.ppd.pok.ibm.com
```

Changing NSD server usage and failback

GPFS determines if a node has physical or virtual connectivity to an underlying NSD disk through a sequence of commands invoked from the GPFS daemon. This determination is called disk discovery and occurs at both initial GPFS startup as well as whenever a file system is mounted.

The default order of access used in disk discovery:

1. Local block device interfaces for SAN, SCSI or IDE disks
2. NSD servers

The **useNSDserver** file system mount option can be used to set the order of access used in disk discovery, and limit or eliminate switching from local access to NSD server access, or the other way around. This option is specified using the **-o** flag of the **mmmount**, **mount**, **mmchfs**, and **mmremotefs** commands, and has one of these values:

always
> Always access the disk using the NSD server.

asfound
> Access the disk as found (the first time the disk was accessed). No change of disk access from local to NSD server, or the other way around, is performed by GPFS.

asneeded
> Access the disk any way possible. This is the default.

never Always use local disk access.

For example, to always use the NSD server when mounting file system **fs1**, issue this command:

```
mmmount fs1 -o useNSDserver=always
```

To change the disk discovery of a file system that is already mounted: cleanly unmount it, wait for the unmount to complete, and then mount the file system using the desired **-o useNSDserver** option.

Enabling and disabling Persistent Reserve

GPFS can use Persistent Reserve (PR) functionality to improve failover times (with some restrictions).

The following restrictions apply to the use of PR:

- PR is supported on both AIX and Linux nodes. However, note the following:
  - If the disks have defined NSD servers, then the NSD server nodes must all be running AIX, or they must all be running Linux.
  - If the disks are SAN-attached to all nodes, then the SAN-attached nodes in the cluster must all be running AIX, or they must all be running Linux.
- The disk subsystems must support PR
- GPFS supports a mix of PR disks and other disks. However, you will only realize improved failover times if **all** disks in the cluster support PR.

- GPFS only supports PR in the local cluster. Remote mounts must access the disks through an NSD server.
- When you enable or disable PR, you must stop GPFS on all nodes.
- Before enabling PR, make sure all disks are in the same initial state.

To enable Persistent Reserve, enter the following command:

```
mmchconfig usePersistentReserve=yes
```

To disable Persistent Reserve, enter the following command:

```
mmchconfig usePersistentReserve=no
```

For fast recovery times with Persistent Reserve, you should also set the *failureDetectionTime* configuration parameter. For fast recovery, a recommended value would be 10. You can set this by issuing the command:

```
mmchconfig failureDetectionTime=10
```

To determine if the disks on the servers and the disks of a specific node have PR enabled, issue the following command from the node:

```
mmlsnsd -X
```

The system responds with something similar to:

```
Disk name  NSD volume ID      Device      Devtype  Node name            Remarks
---------------------------------------------------------------------------------------
gpfs10nsd  09725E5E43035A99   /dev/hdisk6  hdisk   k155n14.kgn.ibm.com  server node,pr=yes
gpfs10nsd  09725E5E43035A99   /dev/hdisk8  hdisk   k155n16.kgn.ibm.com  server node,pr=yes
gpfs10nsd  09725E5E43035A99   /dev/hdisk6  hdisk   k155n17.kgn.ibm.com  directly attached pr=yes
```

If the GPFS daemon has been started on all the nodes in the cluster and the file system has been mounted on all nodes that have direct access to the disks, then **pr=yes** should be on all hdisks. If you do not see this, there is a problem. Refer to the *IBM Spectrum Scale: Problem Determination Guide* for additional information on Persistent Reserve errors.

Chapter 16. Managing protocol services

GPFS provides system administrators with the ability to manage the protocol services.

Configuring and enabling SMB and NFS protocol services

If you have not previously enabled and started the Cluster Export Services (CES) protocol services, enable and start them now.

Prerequisites

When you enable SMB protocol services, the following prerequisites must be met:
* The number of CES nodes must be 16 or lower.
* All CES nodes must be running the same system architecture. For example, mixing nodes based on Intel and Power is not supported.

When you add new CES nodes to a running system where the SMB protocol is enabled, the following prerequisite must be met:
* All CES nodes must be in `SMB HEALTHY` state.

When you remove a CES node from a running system where the SMB protocol is enabled, the following prerequisite must be met:
* All CES nodes (except for the node that is being removed) must be in `SMB HEALTHY` state.

For more information about the SMB states, see `mmces command` in *IBM Spectrum Scale: Command and Programming Reference*.

Enabling protocol services

Issue the following commands to enable SMB and NFS services on all CES nodes:
* `mmces service enable SMB`
* `mmces service enable NFS`

GUI navigation
* To enable SMB services in the GUI, log on to the IBM Spectrum Scale GUI and select **Settings** > **SMB Service**.
* To enable NFS services in the GUI, log on to the IBM Spectrum Scale GUI and select **Settings** > **NFS Service**.

The protocol services that are used need to be started on all CES nodes:
```
mmces service start SMB -a
mmces service start NFS -a
```

After you start the protocol services, verify that they are running by issuing the `mmces state show` command.

Note: The start and stop are maintenance commands. Stopping a service on a particular protocol node without first suspending the node ensures that the public IP addresses on that node stay with that node. In this case, protocol clients that try to connect to the service with these IP addresses fail.

The following sequence of commands demonstrates how to enable and then disable the file data access method, with the NFS service as an example:

1. Issue the following command to enable the service:

   ```
   mmces service enable NFS
   ```

   Note: This command also starts NFS on all CES nodes.

2. Set up the authentication method. The following command specifies the **file** data access method and the **userdefined** authentication type:

   ```
   mmuserauth service create --data-access-method file --type userdefined
   ```

3. Issue the following command to add an export:

   ```
   mmnfs export add gpfs/fs0/fset0
   ```

   where `fs0` is a GPFS file system and `fset0` is an independent fileset.

4. Issue the following commands to verify that the service is configured and running:

   ```
   mmces service list -a
   mmuserauth service list
   mmnfs export list
   ```

5. Issue the following commands to stop NFS and disable the NFS protocol on the CES nodes:

   ```
   mmces service stop nfs -a
   mmuserauth service remove --data-access-method file
   ```

   Note: The sequence for removing the **file** data access method is different for NFS and SMB:
   - For NFS, you must remove the **file** data access method before you disable NFS.
   - For SMB, you cannot remove the **file** data access method while SMB is enabled and running.

6. Issue the following command to disable the NFS service on the CES nodes:

   ```
   mmces service disable NFS
   ```

   Important: When you disable NFS, the NFS configuration is lost. To save the NFS configuration, back up the contents of the `/var/mmfs/ces/nfs-config/` directory on any protocol node.

Configuring and enabling the Object protocol service

If you want to use the Cluster Export Services (CES) object service and it was not configured and enabled during the installation, configure object services now.

1. If a file system for the object data has not been created yet, you must create it now (see **mmcrfs command** in *IBM Spectrum Scale: Command and Programming Reference*).

2. Use the **mmobj** command for the initial configuration of the object stack.

   Note: A separate fileset will be created to hold the object data:

   ```
   mmobj swift base -g /gpfs/fs01 --cluster_hostname clustername --local_keystone --db_password Passw0rd
   --admin_password Password -i 2000 --enable-s3
   ```

   This example creates a fileset for Object storage in the `/gpfs/fs01` fileset with the specified hostname as access point and 2000 inodes. A local database is created for Keystone authentication and S3 API is enabled. See **mmobj command** in *IBM Spectrum Scale: Command and Programming Reference* for details.

3. After the initial configuration, enable and start the object services by running the following commands:

   ```
   mmces service enable OBJ
   ```

4. Verify that the object service is running as expected:

   ```
   mmces service list -a
   ```

Performance tuning for object services

By default, the IBM Spectrum Scale installation sets the number of workers for the object services low. These numbers can be adjusted upwards if you have protocol servers with sufficient cores and memory.

As with most performance tuning, there is no single correct setting. A good starting point for tuning worker counts is to set workers in the `proxy-server.conf` to `auto` so that one worker is started for every core on a protocol node. The other servers can be set to a percentage of the number of cores on your protocol nodes:
- object server set to 75% of core count
- container server set to 50% of core count
- account server set to 25% of core count

Depending on the load of other protocol workloads, the optimal settings for worker count might be higher or lower than this on your system.

For example, if you have 16 cores in your protocol nodes, the following commands can be used to tune your worker settings:

```
mmobj config change --ccrfile proxy-server.conf --section DEFAULT --property workers --value auto

mmobj config change --ccrfile object-server.conf --section DEFAULT --property workers --value 12

mmobj config change --ccrfile container-server.conf --section DEFAULT --property workers --value 8

mmobj config change --ccrfile account-server.conf --section DEFAULT --property workers --value 4
```

Configuring and enabling the BLOCK service

If you have not previously enabled and started the Cluster Export Services (CES) BLOCK service, enable and start it now. The iSCSI protocol in IBM Spectrum Scale is referred to as the BLOCK service.

Prerequisites

All CES nodes must be running the Red Hat Enterprise Linux 7 and later.

The block service is provided as `gpfs.scst` rpm package. After the rpm installation, you need to compile its Linux kernel modules by running **mmbuildgpl** command on each protocol node.

Enabling BLOCK services

To enable the BLOCK service, type:

```
mmces service enable BLOCK
```

The system displays the following prompt message:

```
Block device support in Spectrum Scale is intended for use only in diskless node
remote boot (non-performance-critical), and is not suited for high-bandwidth
block device access needs. Confirm that this matches your use case before enabling
the block service. If you have any questions contact scale@us.ibm.com
Do you want to continue to enable BLOCK service? (yes/no)
```

To continue to enable the BLOCK service, type `Yes` and press `Enter`.

The system displays the following message:

```
c40bbc1xn12.gpfs.net: Loading and configuring SCST
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

The iSCSI protocol must be started on all CES nodes:

```
mmces service start BLOCK -a
```

After you start the BLOCK service, verify that it is running by running the **mmces state show** command.

Note: Start and stop are maintenance commands. Stopping the BLOCK service on a protocol node, without first suspending the node means that the public IP addresses stay with that node even if the BLOCK service is not available on that node. In this event, protocol clients might attempt to connect using these IP addresses and fail to connect to the BLOCK service.

The following example describes the usage of the BLOCK service:

1. Enable and start the BLOCK service:

   ```
   mmces service enable BLOCK
   mmces service start BLOCK
   ```

2. Verify that the BLOCK service is enabled and running:

   ```
   mmces service list
   mmces state show
   ```

3. Define a host name for iSCSI initiator.

   See the hardware and operating system documentation to retrieve the initiator name. The initiator name of the Linux software iSCSI initiator can be found in /etc/iscsi/initiatorname.iscsi.

   ```
   cat /etc/iscsi/initiatorname.iscsi
   InitiatorName=iqn.1994-05.com.redhat:793ad27cb3c
   ```

4. Define the host for the iSCSI initiator as the name of *host1*:

   ```
   mmblock host add --host-alias host1 --iscsi-name iqn.1994-05.com.redhat: 793ad27cb3c
   ```

5. Verify that host is enabled:

   ```
   mmblock host list
   ```

6. Create the volume file.

   Assume that the GPFS file system *gpfs1* is mounted at /gpfs1. Perform one of the following operations:

   - Create a spare file of 30GB volume on *gpfs1* file system:

     ```
     dd if=/dev/zero of=/gpfs1/volumefile1 seek=30000 bs=1M count=1
     ```

   - Create a preallocated file of 30GB volume on *gpfs1* file system:

     ```
     dd if=/dev/zero of=/gpfs1/volumefile1 bs=1M count=30000
     ```

7. Define the volume as *volume1* for the volume file:

   ```
   mmblock volume add --volume-name volume1 -F /gpfs1/volumefile1
   ```

8. Verify that the volume has been defined:

   ```
   mmblock volume list
   ```

9. Map the volume to the iSCSI initiator host:

   ```
   mmblock map-volume add --host-alias host1 --volume-name volume1
   ```

10. Verify that the volume is mapped to the iSCSI initiator host:

    ```
    mmblock map-volume list host1
    ```

The iSCSI volume can be used by the iSCSI initiator host.

See the hardware and operating system documentation to configure the iSCSI initiator. The following uses the Linux software iSCSI initiator running on the Red Hat Enterprise Linux 7 host.

1. Install the iscsi initiator packages:

   ```
   yum install -y iscsi-initiator-utils
   ```

2. Start the iscsi initiator service:

   ```
   systemctl enable iscsid
   ```

```
systemctl start iscsid
```

3. Discover the iSCSI target by specifying the CES IP address as the target portal address:

   Note: You can query CES IP by running the **mmlscluster --ces** command on the GPFS CES node.
   ```
   iscsiadm -m discovery -t st -p 192.168.6.47
   192.168.6.47:3260,1 iqn.1986-03.com.ibm:spectrumscale.192.168.6.47
   ```
4. Open an iSCSI session by logging in to the system:
   ```
   iscsiadm -m node --targetname=iqn.1986-03.com.ibm:spectrumscale.192.168.6.47 --login
   ```
5. List the iSCSI devices:
   ```
   lsscsi --scsi_id
   [113:0:0:0]  disk    SCST_FIO volume1  302  /dev/sdz
   ```

Disabling protocol services

Exercise caution in disabling protocol services so that configuration information is not lost.

To disable a protocol service, issue the **mmces service disable** command with the appropriate service designation:

SMB   Issue the following command:
```
mmces service disable SMB
```

    Note: You can disable SMB only after you remove the authentication method or if the authentication method is **userdefined**.

    Important: Before you disable SMB protocol services, ensure that you save all the SMB configuration information. Disabling SMB protocol services stops SMB on all CES nodes and removes all configured SMB shares and SMB settings. It removes the SMB configuration information from the CCR, removes the SMB clustered databases (tdb's), and removes the SMB-related config files in the /var/mmfs/ces directory on the CES nodes.

    When you re-enable SMB, you must re-create and reconfigure all the SMB settings and exports.

NFS   Issue the following command:
```
mmces service disable NFS
```

    Before you disable NFS protocol services, ensure that you save all the NFS configuration information by backing up the contents of the /var/mmfs/ces/nfs-config/ directory on any CES node. Disabling the NFS service stops NFS on all CES nodes and removes the NFS configuration information from the CCR and from the /var/mmfs/ces/nfs-config/ directory. Previous exports are lost.

OBJ   Issue the following command:
```
mmces service disable OBJ
```

    Note: For more information on disabling the Object services, see "Understanding and managing Object services" on page 237.

Chapter 17. Managing protocol user authentication

The system administrator can configure authentication for both object and file access either during the installation of the system or after the installation. If the authentication configuration is not configured during installation, you can manually do it by using the `mmuserauth service create` command from any node in the IBM Spectrum Scale cluster. This section covers the manual method of configuring authentication for file and object access.

Setting up authentication servers to configure protocol user access

Before you start configuring authentication for protocol access, the system administrator needs to ensure that the authentication server is set up properly and the connection between the IBM Spectrum Scale system and authentication server is established properly.

Depending on the requirement, the IBM Spectrum Scale system administrator needs to set up the following servers:
* Microsoft Active Directory (AD) for file and object access
* Lightweight Directory Access Protocol server for file and object access
* Keystone server to configure local, AD, or LDAP-based authentication for object access. Configuring Keystone is a mandatory requirement if you need to have Object access.

AD and LDAP servers are set up externally. You can configure either an internal or external Keystone server. The installation and configuration of an external authentication server must be handled separately. The IBM Spectrum Scale system installation manages the installation and set up of internal Keystone server.

Integrating with AD server

If the authentication method is selected as AD, the customer must set up the AD server before configuring the authentication method in the IBM Spectrum Scale system.

Ensure that you have the following details before you start configuring AD-based authentication:
* IP address or host name of the AD server.
* Domain details such as the following:
  - Domain name and realm.
  - AD admin user ID and password to join the IBM Spectrum Scale system as machine account into the AD domain.
* ID map role of the system is identified.
* Define the ID map range and size depending upon the maximum RID (sum of allocated and expected growth).
* Primary DNS is added in the `/etc/resolv.conf` file on all the protocol nodes. It resolves the authentication server system with which the IBM Spectrum Scale system is configured. This is a mandatory requirement when AD is used as the authentication server as the DNS must be able to resolve the host domain and its trusted domains of interest. The manual changes done to the configuration files might get overwritten by the Operating System's network manager. So, ensure that the DNS configuration is persistent even after you restart the system. For more information on the circumstances where the configuration files are overwritten, see the corresponding Operating System documentation.
* During the AD join process, a computer account having the same name as the netbios name is searched within the AD domain that will be joined. If the name is not found, a new computer entry is

created in the standard location (CN=Computers). If the user chooses to pre-create computer accounts for IBM Spectrum Scale in the AD domain within a particular organizational unit, the computer account must be created with a valid name and it must be passed as the netbios name while configuring the IBM Spectrum Scale system. After the account is created on the AD server, the system must be joined to the AD domain.

To achieve high-availability, you can configure multiple AD domain controllers. While configuring AD-based authentication, you do not need to specify multiple AD servers in the command line to achieve high-availability. The IBM Spectrum Scale system queries the specified AD server for relevant details and configures itself for the AD-based authentication. The IBM Spectrum Scale system relies on the DNS server to identify the set of available AD servers that are currently available in the environment serving the same domain system.

Integrating with LDAP server

If LDAP-based authentication is selected, ensure that the LDAP server is set up with the required schemas to handle the authentication and ID mapping requests. If you need to support SMB data access, LDAP schema must be extended before configuring the authentication.

Ensure that you have the following details before you start configuring LDAP based authentication:
- Domain details such as base dn, and dn prefixes of groups and users, else default values are used. Default user group suffix is `<ou=Groups, <base dn>` and default user suffix is `ou=People, <base dn>`.
- IP address or host name of LDAP server.
- Admin user ID and password of LDAP server that is used during LDAP simple bind and for LDAP searches.
- The secret key you provided for encrypting/decrypting passwords unless you have disabled prompting for the key.
- NetBIOS name that is to be assigned for the IBM Spectrum Scale system.
- If you need to have secure communication between the IBM Spectrum Scale system and LDAP, the CA signed certificate that is used by the LDAP server for TLS communication must be placed at the specified location in the system.
- If you are using LDAP with Kerberos, create a Kerberos keytab file by using the MIT KDC infrastructure.
- Primary DNS is added in the /etc/resolv.conf file on all the protocol nodes. It resolves the authentication server system with which the IBM Spectrum Scale system is configured. The manual changes done to the configuration files might get overwritten by the Operating System's network manager. So, ensure that the DNS configuration is persistent even after you restart the system. For more information on the circumstances where the configuration files are overwritten, refer the corresponding Operating System documentation.

Setting up LDAP server prerequisites

Before you start configuring the IBM Spectrum Scale system with LDAP server, the following external LDAP server prerequisites must be met:
- The LDAP server must already be configured.
- Enable TLS encryption on the LDAP server, if you need to secure communication between the IBM Spectrum Scale system and LDAP server. Details on configuring SSL or TLS encryption on the server can be obtained from the *OpenLDAP Administrator's Guide*.
- To access SMB shares, LDAP user information must be updated with unique Samba attributes in addition to the attributes that are stored for a normal LDAP user. Ensure that these required Samba attributes are present in the LDAP user entries.
- Ensure you do not have the same user name for different organizational units of the LDAP server that is configured with the IBM Spectrum Scale system.

LDAP bind user requirements:

When an IBM Spectrum Scale system is configured with LDAP as the authentication method, the IBM Spectrum Scale system needs to connect to the LDAP server by using an administrative user ID and password. This administrative user is referred as bind user.

It is recommended that the bind user is given enough privileges that are required by the storage system to mitigate any security concerns.

This bind user must at least have permission to query users and groups that are defined in the LDAP server to allow storage system to authenticate these users. The bind user information (`bind dn`) is also used by the Samba server while making LDAP queries to retrieve required information from the LDAP server.

Note: In the following sections, it is assumed that the user account for the bind user exists in the LDAP directory server. The bind user distinguished name (also known as dn) used in the following examples is `uid=ibmbinduser,ou=people,dc=ldapserver,dc=com`. This name needs to be updated based on the bind user that is used with the IBM Spectrum Scale system.

OpenLDAP server ACLs:

The OpenLDAP server ACLs define the privileges that are required for the bind user.

The following example uses ACLs that are required for the bind user and other type of users for the sake of completeness. It is likely that a corporate directory server has those ACLs configured already and only the entries for bind user need to be merged correctly in the `slapd` configuration file (generally, `/etc/openldap/slapd.conf` file on Linux systems). Follow the ACL ordering rules to ensure that correct ACLs are applied.

```
### some attributes need to be readable so that commands like 'id user',
'getent' etc can answer correctly.
access to attrs=cn,objectClass,entry,homeDirectory,uid,uidNumber,
gidNumber,memberUid
by dn="uid=ibmbinduser,ou=people,dc=ldapserver,dc=com" read
```

```
###The following will not list userPassword when ldapsearch is
performed with bind user.
```

```
### Anonymous is needed to allow bind to succeed and users to
authenticate, should be
a pre-existing entry already.
access to attrs=userPassword
    by dn="uid=ibmbinduser,ou=people,dc=ldapserver,dc=com"  auth
    by self write
    by anonymous auth
    by * none
```

```
### Storage system needs to be able to find samba domain account
specified on the mmuserauth service create command.
```

```
###It is strongly recommended that domain account is pre-created
to ensure
```

```
###consistent access to multiple storage systems.
```

```
###Uncomment ONLY if you want storage systems to create domain
account when it does not exist.
#access to dn.base="dc=ldapserver,dc=com"
#    by dn="uid=ibmbinduser,ou=people,dc=ldapserver,dc=com " write
#    by * none
```

```
access to dn.regex="sambadomainname=[^,]+,dc=ldapserver,dc=com"
    by dn=" uid=ibmbinduser,ou=people,dc=ldapserver,dc=com" read
    by * none

### all samba attributes need to be readable for samba access
access to attrs=cn,sambaLMPassword,sambaNTPassword,sambaPwdLastSet,
sambaLogonTime,sambaLogoffTime,sambaKickoffTime,sambaPwdCanChange,
sambaPwdMustChange,sambaAcctFlags,displayName,sambaHomePath,
sambaHomeDrive,sambaLogonScript,sambaProfilePath,description,
sambaUserWorkstations,sambaPrimaryGroupSID,sambaDomainName,
sambaMungedDial,sambaBadPasswordCount,sambaBadPasswordTime,
sambaPasswordHistory,sambaLogonHours,sambaSID,sambaSIDList,
sambaTrustFlags,sambaGroupType,sambaNextRid,sambaNextGroupRid,
sambaNextUserRid,sambaAlgorithmicRidBase,sambaShareName,
sambaOptionName,sambaBoolOption,sambaIntegerOption,
sambaStringOption,sambaStringListoption
    by dn="uid=ibmbinduser,ou=people,dc=ldapserver,dc=com" read
    by self read
    by * none

### Need write access to record bad failed login attempt
access to attrs=cn,sambaBadPasswordCount,sambaBadPasswordTime,
sambaAcctFlags by dn="uid=ibmbinduser,ou=people,dc=ldapserver,
dc=com" write

### Required to check samba schema
access to dn.base=* by  dn="uid=ibmbinduser,ou=people,
dc=ldapserver,dc=com" read
```

IBM Spectrum Protect Directory Server ACLs:

The IBM Spectrum Protect Directory Server ACLs define the privileges that are required for the bind user, when using IBM Spectrum Protect Directory Server.

These ACLs are provided in the LDIF format and can be applied by submitting the **ldapmodify** command.

```
dn: dc=ldapserver,dc=com
changetype: modify

add: ibm-filterAclEntry

ibm-filterAclEntry:access-id:uid=ibmbinduser,ou=people,dc=ldapserver,dc=com:
(objectClass=sambaSamAccount):normal:rsc:sensitive:rsc:critical:rsc
-
add:ibm-filterAclEntry

ibm-filterAclEntry:access-id:uid=ibmbinduser,ou=people,dc=ldapserver,dc=com:
(objectclass=sambaDomain):normal:rwsc:sensitive:rwsc:critical:rwsc

 dn:uid=ibmbinduser,ou=people,dc=ldapserver,dc=com

add:aclEntry

aclentry: access-id:uid=ibmbinduser,ou=people,dc=ldapserver,dc=com:at.cn:r:at.
objectClass:r:at.homeDirectory:r:at.uid:r:at.uidNumber:s:

at.gidNumber:r:at.memberUid:r:at.userPassword:sc:at.sambaLMPassword:r:at.
sambaNTPassword:r:at.sambaPwdLastSet:r:at.sambaLogonTime:r:

at.sambaLogoffTime:r:at.sambaKickoffTime:r:at.sambaPwdCanChange:r:at.
sambaPwdMustChange:r:at.sambaAcctFlags:r:at.displayName:r:
```

```
at.sambaHomePath:r:at.sambaHomeDrive:r:at.sambaLogonScript:r:at.sambaProfilePath:
r:at.description:r:at.sambaUserWorkstations:r:

at.sambaPrimaryGroupSID:r:at.sambaDomainName:r:at.sambaMungedDial:r:at.
sambaBadPasswordCount:r:at.sambaBadPasswordTime:r:
at.sambaPasswordHistory:r:at.sambaLogonHours:r:at.sambaSID:r:at.sambaSIDList:r:at.
sambaTrustFlags:r:at.sambaGroupType:r:
at.sambaNextRid:r:at.sambaNextGroupRid:r:at.sambaNextUserRid:r:at.
sambaAlgorithmicRidBase:r:at.sambaShareName:r:at.sambaOptionName:r:

at.sambaBoolOption:r:at.sambaIntegerOption:r:at.sambaStringOption:r:at.
sambaStringListoption:r:at.sambaBadPasswordCount:rwsc:

at.sambaBadPasswordTime:rwsc:at.sambaAcctFlags:rwsc

### Storage system needs to be able to find samba domain account specified
on the mmuserauth service create command.

###It is strongly recommended that domain account is pre-created to ensure

###consistent access to multiple storage systems.

###Uncomment ONLY if you want storage systems to create domain account when
it does not exist.

 dn: dc=ldapserver,dc=com

changetype: modify

add:ibm-filterAclEntry

ibm-filterAclEntry:access-id:uid=ibmbinduser,ou=people,dc=ldapserver,
dc=com:(objectclass=domain):object:grant:a
```

See *IBM Tivoli Directory Server Administration Guide* for information about applying these ACLs on the IBM Spectrum Protect Directory Server.

Updating LDAP user information with Samba attributes

If you need to support SMB data access, LDAP schema must be extended to store more attributes such as SID, Windows password hash to the POSIX user object. To use Samba accounts, update LDAP user information with unique Samba attributes.

The following sample LDIF file shows the minimum required samba attributes:

```
dn: cn=SMBuser,ou=People,dc=ibm,dc=com
changetype: modify
add : objectClass
objectClass: sambaSamAccount
-
add: sambaSID
sambaSID: S-1-5-21-1528920847-3529959213-2931869277-1102
-
add:sambaPasswordHistory
sambaPasswordHistory: 00000000000000000000000000000000000000000000000000000000
-
add:sambaNTPassword
sambaNTPassword: (valid samba password hash )
-
add:sambaPwdLastSet
```

```
sambaPwdLastSet: 1263386096
-
add:SambaAcctFlags
sambaAcctFlags: [U         ]
```

Note: Attributes must be separated with a dash as the first and only character on a separate line.

Perform the following steps to create the values for `sambaNTPassword`, `sambaPwdLastSet`, and `SambaAcctFlags`, which must be generated from a PERL module:

1. Download the module from http://search.cpan.org/~bjkuit/Crypt-SmbHash-0.12/SmbHash.pm. Create and install the module by following the readme file.

2. Use the following PERL script to generate the LM and NT password hashes:

```
# cat /tmp/Crypt-SmbHash-0.12/gen_hash.pl
#!/usr/local/bin/perl
use Crypt::SmbHash;
$username = $ARGV[0];
$password = $ARGV[1];
if ( !$password ) {
        print "Not enough arguments\n";
        print "Usage: $0 username password\n";
        exit 1;
}
$uid = (getpwnam($username))[2];
my ($login,undef,$uid) = getpwnam($ARGV[0]);
ntlmgen $password, $lm, $nt;
printf "%s:%d:%s:%s:[%-11s]:LCT-%08X\n", $login, $uid, $lm, $nt, "U", time;
```

3. Generate the password hashes for any user as in the following example for the user test01:

```
# perl gen_hash.pl SMBuser test01

:0:47F9DBCCD37D6B40AAD3B435B51404EE:82E6D500C194BA5B9716495691FB7DD6:
[U         ]:LCT-4C18B9FC
```

Note: The output contains login name, uid, LM hash, NT hash, flags, and time, with each field separated from the next by a colon. The login name and uid are omitted because the command was not run on the LDAP server.

4. Use the information from step 3 to update the LDIF file in the format that is provided in the example at the beginning of this topic.
   - To generate the `sambaPwdLastSet` value, use the hexadecimal time value from step 3 after the dash character and convert it into decimal.
   - A valid samba SID is required for a user to enable that user's access to an IBM Spectrum Scale share. To generate the samba SID, multiply the user's UID by 2 and add 1000. The users's SID must contain the samba SID from the `sambaDomainName`, which is either generated or picked up from the LDAP server, if it exists. The following attributes for `sambaDomainName` LDIF entry are required:

```
dn: sambaDomainName=(IBM Spectrum Scale system),dc=ibm,dc=com
sambaDomainName: (IBM Spectrum Scale system name)
sambaSID: S-1-5-21-1528920847-3529959213-2931869277
sambaPwdHistoryLength: 0
sambaMaxPwdAge: -1
sambaMinPwdAge: 0
```

This entry can be created by the LDAP server administrator by using either of the following two methods:

   - Write and run a bash script similar to the following example:

```
sambaSID=
        for num in 1 2 3 ;do
            randNum=$(od -vAn -N4 -tu4 < /dev/urandom | sed -e 's/ //g')
            if [ -z     "$sambaSID" ];then
                sambaSID="S-1-5-21-$randNum"
            else
```

```
                    sambaSID="${sambaSID}-$      {randNum}"
              fi
          done
      echo $sambaSID
```

Then, use the samba SID generated to create the LDIF file. The `sambaDomainName` must match the IBM Spectrum Scale system name.

  – When you run the **mmuserauth service create** command, the system creates the `sambaDomainName`, if it does not exist.

The `sambaSID` for every user must have the following format: (samba SID for the domain)-(userID*2+1000). For example: `S-1-5-21-1528920847-3529959213-2931869277-1102`

Note: To enable access using the same LDAP server domain to more than one IBM Spectrum Scale system or other IBM NAS like IBM SONAS or IBM V7000 Unified, the Samba domain SID prefix of all of the systems must match. The Samba domain SID prefix is used to prepare the SID of users/groups planning to access the IBM Spectrum Scale system via CIFS. So, if you change the Samba domain SID for an IBM Spectrum Scale system on the LDAP server, you must restart the CES Samba service on that IBM Spectrum Scale system for the change to take effect.

5. Submit the **ldapmodify** command as shown in the following example to update the user's information :

```
# ldapmodify -h localhost -D cn=Manager,dc=ibm,dc=com -W -x -f /tmp/samba_user.ldif
```

Integrating with Keystone Identity Service

The object protocol uses the keystone service to authenticate object users. When configuring the IBM Spectrum Scale system, you must specify that either an internal keystone server or an external keystone server will be used. In either case, the keystone server can use a local database or a separate LDAP or AD system for managing user credentials. If you are using an external keystone server, you are responsible for the configuration of this service. For more information, refer to the OpenStack documentation.

Before you configure authentication for object, ensure that the object services are enabled. To enable object services, use the **mmces service enable obj** command.

Prerequisites

Ensure that you have the following details before you start configuring local authentication for object access:

• The keystone host name must be defined and configured on all protocol nodes of the cluster. This host name returns one of the CES IP addresses, such as a round-robin DNS. It could also be a fixed IP of a load balancer that distributes requests to one of the CES nodes. This host name is also used to create the keystone endpoints.

• If you want the keystone server to use external CA signed certificate for signing the keystone generated tokens, ensure that the following certificates are available in /var/mmfs/tmp directory of the node where you run the commands:

  – /var/mmfs/tmp/signing_cert.pem
  – /var/mmfs/tmp/signing_key.pem
  – /var/mmfs/tmp/signing_cacert.pem

Note: By default, the system uses an internal self-signed certificate that is generated by **keystone-manage pki_setup**. For production environments we recommend the use of certificates issued by a trusted CA rather than self-signed certificates.

• If you want to enable SSL on keystone, ensure that the following certificates that are placed on the /var/mmfs/tmp directory of the node where commands are run:

  – /var/mmfs/tmp/ssl_cert.pem
  – /var/mmfs/tmp/ssl_key.pem

- /var/mmfs/tmp/ssl_cacert.pem
- /var/mmfs/tmp/ks_ext_cacert.pem

Note: By default, the IBM Spectrum Scale installation process configures object authentication with a local keystone authentication method.

Configuring authentication and ID mapping for file access

The system administrator can decide whether to configure authentication and ID mapping method either during the installation of the IBM Spectrum Scale system or after the installation. If the authentication configuration is not configured during installation, you can manually do it by using the **mmuserauth service create** command from any protocol node of the IBM Spectrum Scale system. This section covers the manual method of configuring authentication for file access.

You can configure the following external authentication servers for file access:
- Active Directory (AD)
- Light Weight Directory Access Protocol (LDAP)
- Network Information Service (NIS)

Before you configure the authentication method, ensure that the following RPMs are installed on all the protocol nodes before you start configuring the authentication method:

Note: If you try to configure the file authentication method manually, with the **mmuserauth cli** command, the command displays an error message if the required RPMs are not installed on the nodes. The error output includes a list of nodes in which some RPMs are not installed and a list of the missing RPMs for each node.

On Red Hat Enterprise Linux nodes
- **For AD:**
  - `bind-utils`
  - `krb5-workstation`
- **For LDAP:**
  - `openldap-clients`
  - `sssd` and its dependencies (particularly `sssd-common` and `sssd-ldap`). It is a good idea to install all the dependencies, as in the following example:
    `yum install sssd*`
  - `krb5-workstation` only if Kerberized authentication is planned.
- **For NIS:**
  - `sssd` and its dependencies (particularly `sssd-common` and `sssd-proxy`)
  - `ypbind` and its dependencies (`yp-tools`)

On SLES nodes
- **For AD:**
  - `bind-utils`
  - `krb5-client`
- **For LDAP:**
  - `openldap2-client`
  - `sssd` and its dependencies (particularly `sssd-common`, `sssd-ldap`, and `sssd-krb5`). It is a good idea to install all the dependencies, as in the following example:
    `zypper install sssd*`
  - `krb5-client` only if Kerberized authentication is planned.

- **For NIS:**
  - sssd and its dependencies (particularly `sssd-common` and `sssd-proxy`)
  - ypbind and its dependencies (`yp-tools`)

On Ubuntu 16 nodes

- **For AD:**
  - dnsutils
  - krb5-user (only if Kerberos authentication is planned)
- **For LDAP:**
  - ldap-utils
  - krb5-user (only if Kerberos authentication is planned)
  - sssd and its dependencies. It is a good idea to install all the dependencies, as in the following example:
    ```
    apt-get install sssd
    ```
- **For NIS:**
  - sssd and its dependencies (particularly `sssd-common` and `sssd-proxy`)
  - nis and `libslp1` (`nis` package automatically installs the `libslp1` package)

Prerequisite for configuring Kerberos-based SMB access

The following requirements must be met to configure IBM Spectrum Scale for Kerberized SMB access:

- The time must be synchronized across the KDC server, the IBM Spectrum Scale cluster protocol nodes, and the SMB clients, or else access to an SMB share could be denied.
- In MIT KDC configurations for the SMB services, the service principal name must use the NetBIOS name and the realm name. For example, if the NetBIOS name is FOO and the realm is KDC.COM, the service principal name should be cifs/foo@KDC.COM. The NetBIOS name is the value specified for the option --netbios_name in the **mmuserauth** command. The realm may be discovered from the value stored for Alt_Name returned from the command: wbinfo -D <domain>.
- The clients should use only the NetBIOS name when accessing an SMB share. Using any other name or IP address might either cause a failure to connect or fallback to NTLM authentication.
- With Active Directory KDC, you can use DNS alias (CNAME) for Kerberized SMB access. To use the alias, you must register the DNS alias (CNAME) record for the NetBIOS name (system account name) using the SetSPN tool available on Active Directory server. For example, if the NetBIOS name is FOO and the DNS alias is BAR, use the SetSPN tool from the command prompt of the Active Directory server to register the record, "setspn -A cifs/BAR FOO". Not registering the DNS alias record for the NetBIOS name might cause access to the SMB shares to be denied with the error code, KDC_ERR_S_SPRINCIPAL_UNKNOWN.
- On Linux clients, to use Kerberized SMB access for IBM Spectrum Scale configured with MIT KDC, you must at least have the 3.5.9 version of Samba client installed. The Linux clients having an older Samba client might encounter the following error, while trying to access SMB shares:

```
ads_krb5_mk_req: krb5_get_credentials failed for foo$@KDC.COM (Server not found in Kerberos database)
 cli_session_setup_kerberos: spnego_gen_negTokenTarg failed: Server not found in Kerberos database
```

To determine if a client has authenticated via Kerberos, either verify at the client or collect a protocol trace:

```
mmprotocoltrace start smb -c x.x.x.x
```

Replace x.x.x.x with the IP address of the client system access IBM Spectrum Scale to be verified.

Access the IBM Spectrum Scale SMB service from that client.

Then, issue the command:

```
mmprotocoltrace stop smb
```

Extract the compressed trace files and look for the file ending with smbd.log. If that file contains an entry similar to "Kerberos ticket principal name is..." then Kerberos is being used.

Note: It is not recommended to run for extended periods of time at log levels higher than 1 as this could impact performance.

Configuring AD-based authentication for file access

You can configure Microsoft Active Directory (AD) as the authentication server to manage the authentication requests and to store user credentials.

You can configure AD-based authentication with the following ID mapping methods:
* RFC2307
* Automatic
* LDAP

RFC2307 ID mapping

In the RFC2307 ID mapping method, the user and group IDs are stored and managed in the AD server and these IDs are used by the IBM Spectrum Scale system during file access. The RFC2307 ID mapping method is used when you want to have multiprotocol access. That is, you can have both NFS and SMB access over the same data.

Automatic ID mapping

In the automatic ID mapping method, user ID and group ID are automatically generated and stored within the IBM Spectrum Scale system. When an external ID mapping server is not present in the environment or cannot be used, then this ID mapping method can be used. This method is typically used if you have SMB only access and do not plan to deploy multiprotocol access. That is, the AD-based authentication with automatic ID mapping is not used if you need to allow NFS and SMB access to the same data.

LDAP ID mapping

In the LDAP mapping method, user ID and group ID are stored and managed in the LDAP server, and these IDs are used by the IBM Spectrum Scale system during file access. The LDAP ID mapping method is used when you want to have multiprotocol access. That is, you can have both NFS and SMB access over the same data.

Setting up a range of ID maps that can be allotted to the users

You can optionally specify the pool of values from which the UIDs and GIDs are assigned by the IBM Spectrum Scale system to Active Directory users and groups. When a user or group is defined in AD, it is identified by a security identifier (SID), which includes a component that is called Relative Identifier (RID). The RID value depends on the number of users and groups in the Active Directory domain. The `--idmap-range` and `--idmap-range-size` parameters of the `mmuserauth service create` command specify the pool from which UIDs and GIDs are assigned by the IBM Spectrum Scale system to AD users and group of users.

The ID map range is defined between a minimum and maximum value. The default minimum value is 10000000 and the default maximum value is 299999999, and the default range size is 1000000. This allows for a maximum of 290 unique Active Directory domains.

The ID map range size specifies the total number of UIDs and GIDs that are assignable per domain. For example, if range is defined as 10000-20000, and range size is defined as 2000 (`--idmap-range 10000-20000:2000`), five domains can be mapped, each consisting of 2000 IDs. Ensure that range size is defined such that at least three domains can be mapped. The range size is identical for all AD domains that are configured by the IBM Spectrum Scale system. Choose an ID map range size that allows for the highest anticipated RID value among all of the anticipated AD users and group of users in all of the anticipated AD domains. Ensure that the range size value, when originally defined, takes into account the

planned growth in the number of AD users and groups of users. The ID map range size cannot be changed after the IBM Spectrum Scale system is configured with Active Directory as the authentication server.

Whenever a user or user group from an AD domain accesses the IBM Spectrum Scale system, a range is allocated per domain. UID or GID for a user or user group is allocated depending upon this range and the RID of the user or user group. If RID of any user or group is greater than the range size, then that user or user group is mapped into extension ranges depending upon the number of available ranges. If the number of ranges (default value is 290) runs out, then mapping requests for a new user or user group (or new extension ranges for user and group that is already known) are ignored and thus that user and user group cannot access the data.

Choosing range size

1. Determine the highest Active Directory RID that is currently assigned. You can use the **dcdiag** command at the command prompt of the operating system of the server that is hosting Active Directory to determine the value of the rIDNextRID attribute. For example:

   ```
   # dcdiag /s:IP_of_system_hosting_AD  /v /test:ridmanager
   ```

   Specifically,

   ```
   C:\Program Files\Support Tools>dcdiag /s:10.0.0.123 /v /test:ridmanager
   ```

   The following output is displayed:

   ```
   Starting test: RidManager
       * Available RID Pool for the Domain is 1600 to 1073741823
       * win2k8.pollux.com is the RID Master
       * DsBind with RID Master was succesRFC2307l
       * rIDAllocationPool is 1100 to 1599
       * rIDPreviousAllocationPool is 1100 to 1599
       * rIDNextRID: 1174
   ```

   In this example, the rIDNextRID value is 1174. Another way to determine the current value for rIDNextRID is to run an LDAP query on the following DN Path:

   ```
   CN=Rid Set,Cn=computername,ou=domain controllers,DC=domain,DC=COM
   ```

   If there is more than one domain controller serving the Active Directory domain, determine the highest RID among all of the domain controllers. Similarly, if there is more than one domain, determine the highest RID among all of the domains.
2. Estimate the expected number of users and groups that might be added in future, in addition to the current number of users and groups.
3. Add the highest RID determined in step 1 to the number of users and groups that were estimated in the previous step. The result is the estimate for the value of the range size.

Considerations for changing the ID map range and range size

If the IBM Spectrum Scale system is configured to use AD-based authentication, only the maximum value of ID map range can be changed. All other changes to ID map range and size, except increasing the maximum value of ID map range require reconfiguration of authentication, which results in loss of access to data. For example, if you used the **--idmap-range** as 3000-10000 and **--idmap-range-size** as 2000, you can increase only the value 10000 to accommodate more users per domain, without having an impact on the access to the data.

To change the ID mapping of an existing AD-based authentication configuration, either to change the range minimum value, decrease the range maximum value, or change the range size, you must complete the following steps:

Note: The **mmuserauth service remove** command results in loss of access.

1. Submit the **mmuserauth service remove** command and do not specify the **--idmapdelete** option.
2. Submit the **mmuserauth service remove** command and do specify the **--idmapdelete** option.

3. Submit the `mmuserauth service create` command with the options and values that you want for the new Active Directory configuration.

   Important: If you do not perform the preceding three steps in sequence, results are unpredictable and can include complete loss of data access.

Prerequisite for configuring AD-based authentication for file access

See "Integrating with AD server" on page 175 for more information on the prerequisites for integrating AD server with the IBM Spectrum Scale system.

You need to run the `mmuserauth service create` command with the following mandatory parameters to create AD based authentication for file access:

- --type ad
- --data-access-method file
- --servers *<server host name or IP address>*
- --netbios-name *<netBiosName>*
- --user-name *<admin-username>*
- --password *<admin-password>*. This is optional while entering the parameters but the system prompts you to enter the password when you run the command.
- --unixmap-domains <unixDomainMap>. This option is mandatory if RFC2307 ID mapping is used. For example, `--unixmap-domains DOMAINS(5000-20000)`. Specifies the Active Directory domains for which user ID and group ID should be fetched from the Active directory server (RFC2307 schema attributes)
- --idmap-role `master` | `subordinate`. While using automatic ID mapping, in order to have same ID maps on systems sharing Active File Manager (AFM) relationship, you need to export the ID mappings from the system whose ID map role is master to the system whose ID map role is subordinate.

See the `mmuserauth service create` command for more information on each parameter.

Prerequisites for configuring AD with RFC2307

The following prerequisites are specific to AD with RFC2307 configuration:

- RFC2307 schema is extended on the AD and all UNIX attributes (including UID and GID) are populated.
- If a trusted domain is configured with ID mapping from RFC2307, the trusted domain must have two-way trust with the host domain, which is the Active Directory domain that is configured for use with the IBM Spectrum Scale system. For example, assume that there are three domains in trusted relationship, X, Y, Z, and that the IBM Spectrum Scale system is configured with domain X as the host domain. If RFC2307 ID mappings are required for domains Y and Z, domains Y and Z must each have a two-way trust with the domain X. `X <-> Y ; X <-> Z`.
- User and group in the Active Directory domain, configured with ID mapping from RFC2307, must have a valid UID and a valid GID assigned to enable access to IBM Spectrum Scale system exports. The UID and GID number that is assigned must be within the ID map range that is specified in the `mmuserauth service create` command. Any users or groups from this domain that do not have UID or GID attributes configured are denied access.

Note: The primary Windows group that is assigned to an AD user must have a valid GID assigned within the specified ID mapping range. The primary Windows group is usually located in the Member Of tab in the user's properties. The primary Windows group is different from the UNIX primary group, which is listed in the UNIX Attributes tab. A user is denied access if that user's Windows primary group does not have a valid GID assigned. The UNIX primary group attribute is ignored.

In case of a mutual trust setup between two independent AD domains, DNS forwarding must be configured between the two trust.

Configuring AD-based authentication with automatic ID mapping

When the IBM Spectrum Scale system is configured for AD-based authentication, automatic ID mapping method can be used to create UID or GID of a user or group respectively. The ID maps are stored within the IBM Spectrum Scale system.

The following provides an example of how to configure an IBM Spectrum Scale system with Active Directory and automatic ID mapping.

1. Issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ad --data-access-method file --netbios-name
ess --user-name administrator --idmap-role master --servers myADserver  --idmap-range-size 1000000
--idmap-range 10000000-299999999
```

   The system displays the following output:

```
File authentication configuration completed successfully.
```

2. Verify the authentication configuration by issuing the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

   The system displays the following output:

```
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     false
SERVERS                 myADserver
USER_NAME               administrator
NETBIOS_NAME            ess
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         none
LDAPMAP_DOMAINS         none

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

3. Verify the user resolution on the system:

```
# id "DOMAIN\\user1"
uid=12001172(DOMAIN\user1) gid=12001174(DOMAIN\group1) groups=12001174
(DOMAIN\group1),12001172(DOMAIN\user1),12000513(DOMAIN\domain users),
11000545(BUILTIN\users)
```

Configuring AD-based authentication with RFC2307 ID mapping

You can configure IBM Spectrum Scale system authentication with Active Directory (AD) and RFC2307 and Active Directory (AD) with Kerberos and RFC2307 ID mapping. In these authentication methods, use Active Directory to store user credentials and RFC2307 server to store UIDs and GIDs. This is useful when you are planning to use any pre-existing UNIX client or NFS and SMB protocols for data access with the AFM feature of the IBM Spectrum Scale system. If you use AD-based authentication and the ID maps are not configured with RFC2307, the IBM Spectrum Scale system uses the automatic ID mappings by default.

The following provides an example of configuring AD with RFC2307 ID mapping:

1. Submit the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create  --type ad --data-access-method file --netbios-name
ess --user-name administrator --idmap-role master --servers myADserver
 --idmap-range-size 1000000 --idmap-range 10000000-299999999
--unixmap-domains 'DOMAIN(5000-20000)'
```

   The system displays the following output:

```
File authentication configuration completed successfully.
```

2. Issue the **mmuserauth service list** to verify the authentication configuration as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     false
SERVERS                 myADserver
USER_NAME               administrator
NETBIOS_NAME            ess
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         DOMAIN(5000-20000)
LDAPMAP_DOMAINS         none

OBJECT access not configured
PARAMETERS              VALUES
```

3. Verify the user name resolution on the system. Confirm that the resolution is showing IDs that are pulled from RFC2307 attributes on the AD server.

```
# id DOMAIN\\administrator
uid=10002(DOMAIN\administrator) gid=10000(DOMAIN\domain users)
groups=10000(DOMAIN\domain users
```

Configuring AD using Kerberos with RFC2307 ID mapping:

1. Submit the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create  --data-access-method file --type ad --netbios-name
kknode_v42 --servers myADserver --user-name administrator  --idmap-role master
--enable-nfs-kerberos --unixmap-domains "DOMAIN(10000-200000)"
```

The system displays the following output:

```
File authentication configuration completed successfully.
```

2. Issue the **mmuserauth service list** to verify the authentication configuration as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     true
SERVERS                 myADserver
USER_NAME               administrator
NETBIOS_NAME            kknode_v42
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         DOMAIN(1000-200000)
LDAPMAP_DOMAINS         none

OBJECT access not configured
PARAMETERS              VALUES
```

3. Verify the user name resolution on the system. Confirm that the resolution is showing IDs that are pulled from RFC2307 attributes on the AD server.

```
# id DOMAIN\\administrator
uid=10002(DOMAIN\administrator) gid=40000(DOMAIN\domain users)
groups=11000545(BUILTIN\users),11000544 (BUILTIN\administrators)
```

AD authentication with RFC2307 ID mapping for picking UNIX primary group:

You can configure IBM Spectrum Scale system authentication with Active Directory (AD) and RFC2307 ID mapping or Active Directory (AD) with Kerberos NFS and RFC2307 ID mapping. In these authentication methods, use Active Directory to store user credentials and RFC2307 attributes on the same AD server to store UIDs and GIDs. These authentication schemes are useful when you are planning to use any pre-existing UNIX client or NFS protocol together with SMB protocols for data access. RFC2307 ID mapping is configurable per AD domain. If you use AD-based authentication and the ID maps are not configured with RFC2307, the IBM Spectrum Scale system uses the automatic ID mappings by default.

The following provides an example of how to configure the IBM Spectrum Scale system with Active Directory and RFC2307 ID mapping for picking UNIX primary group:

1. Submit the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create  --type ad --data-access-method file --netbios-name ess
--user-name administrator --idmap-role master --servers myADserver  --idmap-range-size 1000000
 --idmap-range 10000000-299999999  --unixmap-domains 'DOMAIN(5000-20000:unix)'
```

   The system displays this output:

```
File authentication configuration completed successfully.
```

2. Issue this command to verify the authentication configuration:

```
mmuserauth service list
```

   The system displays the following output:

```
FILE access configuration : AD
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS      false
SERVERS                  myADserver
USER_NAME                administrator
NETBIOS_NAME             ess
IDMAP_ROLE               master
IDMAP_RANGE              10000000-299999999
IDMAP_RANGE_SIZE         1000000
UNIXMAP_DOMAINS          DOMAIN(5000-20000:unix)
LDAPMAP_DOMAINS          none

OBJECT access not configured
PARAMETERS               VALUES
```

3. Verify the user name resolution on the system after successfully authenticating the user. Confirm that the resolution is showing primary group picked up as defined in UNIX attribute of the user. Validate the IDs that are pulled are from RFC2307 attributes on the AD server:

```
# id DOMAIN\\unixuser
```

   The system displays the following output:

```
uid=10002(DOMAIN\unixuser) gid=10000(DOMAIN\unix users)
groups=10000(DOMAIN\unix users), 11000545(BUILTIN\users),11000544 (BUILTIN\administrators)
```

Configuring AD using Kerberos with RFC2307 ID mapping

1. Submit the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --data-access-method file --type ad --netbios-name ess --servers
myADserver --user-name administrator --idmap-role master --enable-nfs-kerberos --unixmap-domains
"DOMAIN(10000-200000:unix)"
```

The system displays the following output:

```
File authentication configuration completed successfully.
```

2. Issue the **mmuserauth service list** command to verify the authentication configuration as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     true
SERVERS                 myADserver
USER_NAME               administrator
NETBIOS_NAME            ess
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         DOMAIN(1000-200000:unix)
LDAPMAP_DOMAINS         none

OBJECT access not configured
PARAMETERS              VALUES
```

3. Verify the user name resolution on the system after successfully authenticating the user. Confirm that the resolution is showing primary group picked up as defined in the UNIX attribute of the user. Validate the IDs that are pulled are from RFC2307 attributes on the AD server:

```
# id DOMAIN\\unixuser
```

The system displays the following output:

```
uid=10002(DOMAIN\unixuser) gid=10000(DOMAIN\unix users)
groups=10000(DOMAIN\unix users), 11000545(BUILTIN\users),11000544 (BUILTIN\administrators)
```

Best practices for configuring AD with RFC2307 as the authentication method:

It is recommended to adhere to the following best practices if you configure AD with RFC2307 as the authentication method:

- Remove any internal ID mappings present in the system before you configure AD with RFC2307. Otherwise, the system might detect the internal ID mappings instead of the RFC2307 ID mapping and abort the operation with an error message. In such situations, you are expected to clean up the entire authentication and ID mapping by using the **mmuserauth service remove** and **mmuserauth service remove --idmapdelete** command and then reconfigure AD authentication and RFC2307 ID mapping.
- If data is already present on the system, a complete removal of the authentication and ID mapping can cause permanent loss of data access.
- Using UIDs and GIDs greater than 1000 can avoid an overlap of IDs used by end users, administrative users, and operating system component users of the IBM Spectrum Scale system.

You can use AD-based authentication and RFC2307 ID mapping if you want to use the AFM feature of the IBM Spectrum Scale system.

Limitations of the mmuserauth service create command while configuring AD with RFC2307:

The **mmuserauth service create** command that is used to configure authentication has the following limitations:

- The **mmuserauth service create** command does not check the two-way trust between the host domain and the RFC2307 domain that is required for ID mapping services to function properly. The customer is responsible for configuring the two-way trust relationship between these domains.

- The customer is responsible for installing RFC2307 on the desired AD server, and for assigning UIDs to users and GIDs to groups. The command does not return an error if RFC2307 is not installed, or if a UID or GID is not assigned.

Configuring AD-based authentication with LDAP ID mapping

AD authentication with LDAP ID mapping provides a way for IBM Spectrum Scale to read ID mappings from an LDAP server as defined in RFC 2307. The LDAP server must be a stand-alone LDAP server. Mappings must be provided in advance by the administrator by creating the user accounts in the AD server and the posixAccount and posixGroup objects in the LDAP server. The names in the AD server and in the LDAP server have to be the same. This ID mapping approach allows the continued use of existing LDAP authentication servers that store records in the RFC2307 format. The group memberships defined in the AD server are also be honored in the system.

In the following example, AD is configured with LDAP ID mapping.

1. Submit the **mmuserauth service create** command as shown in the following example:

```
mmuserauth service create --data-access-method file --type ad --servers myADserver
--user-name administrator  --netbios-name specscale
--idmap-role master --ldapmap-domains "DOMAIN1(type=stand-alone:range=1000-100000
:ldap_srv=myLDAPserver:usr_dn=ou=People,dc=example,dc=com:grp_dn=ou=Groups,dc=example,
dc=com:bind_dn=cn=manager,dc=example,dc=com:bind_dn_pwd=password)"
```

   Note: The **bind_dn_pwd** cannot contain the following special characters: semicolon (;), colon (:), opening brace '(', or closing brace ')'.
   The system displays the following output:

```
File authentication configuration completed successfully.
```

2. Issue the **mmuserauth service list** to verify the authentication configuration as shown in the following example:

```
# mmuserauth service list
```

   The system displays the following output:

```
FILE access configuration : AD
PARAMETERS VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS false
SERVERS myADserver
USER_NAME administrator
NETBIOS_NAME specscale
IDMAP_ROLE master
IDMAP_RANGE 10000000-299999999
IDMAP_RANGE_SIZE 1000000
UNIXMAP_DOMAINS none
LDAPMAP_DOMAINS DOMAIN1(type=stand-alone: range=1000-100000:
ldap_srv=myLDAPserver:usr_dn=ou=People,dc=example,dc=com:
grp_dn=ou=Groups,dc=example,dc=com:bind_dn=cn-manager,dc=example,dc=com)
```

3. Verify the user name resolution on the system. Confirm that the resolution is showing IDs that are pulled from LDAP attributes on the AD server.

```
# id DOMAIN\\administrator
uid=10002(DOMAIN\administrator) gid=10000(DOMAIN\domain users)
groups=10000(DOMAIN\domain users
```

Configuring LDAP-based authentication for file access

Using LDAP-based authentication can be useful when you use an external LDAP server to store user information and user passwords. In this authentication method, you can use LDAP as the authentication as well as the ID mapping server for both NFS and SMB. Appropriate SMB schema needs to be uploaded in the LDAP if you plan to have SMB access.

Based on the level of security, the following configurations are possible:

- LDAP with TLS
- LDAP with Kerberos
- LDAP with TLS and Kerberos
- LDAP

Using LDAP with TLS secures the communication between the IBM Spectrum Scale system and the LDAP server, assuming that the LDAP server is configured for TLS.

You can use LDAP with Kerberos for higher security reasons. Kerberos is a network authentication protocol that provides secured communication by ensuring passwords are not sent over the network to the system. LDAP with Kerberos is typically used where an MIT KDC infrastructure exists and you are using it for various Kerberized application or if you want to have NFS and SMB with Kerberized access for higher security reasons.

The LDAP server might need to handle the login requests and ID mapping requests from the client that uses SMB protocol. Usually, the ID mapping requests are cached and they do not contribute to the load on the LDAP server unless the ID mapping cache is cleared due to a maintenance action. If the LDAP server cannot handle the load or a high number of connections, then the response to the login requests is slow or it might time out. In such cases, users need to retry their login requests.

It is assumed that LDAP server is set up with the required schemas installed in it to handle the authentication and ID mapping requests. If you need to support SMB data access, LDAP schema must be extended to enable storing of additional attributes such as SID, Windows password hash to the POSIX user object.

Note: The IBM Spectrum Scale system must not be configured with any authentication method before using LDAP as the authentication system for file access.

See "Integrating with LDAP server" on page 176 for more information on the prerequisites for integrating LDAP server with the IBM Spectrum Scale system.

Configuring LDAP with TLS for file access

You can configure LDAP with TLS as the authentication method for file access. Using TLS with LDAP helps you to have a secure communication channel between the IBM Spectrum Scale system and LDAP server.

In the following example, LDAP is configured with TLS as the authentication method for file access.

1. Ensure that the CA certificate for LDAP server is placed under /var/mmfs/tmp directory with the name ldap_cacert.pem; specifically, on the protocol node where the command is run. Perform validation of CA cert availability with desired name at required location as shown in the following example:

```
# stat /var/mmfs/tmp/ldap_cacert.pem
File: /var/mmfs/tmp/ldap_cacert.pem
Size: 2130 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 12:37:34.088837381 +0530
Modify: 2015-01-23 12:16:24.438837381 +0530
Change: 2015-01-23 12:16:24.438837381 +0530
```

2. Issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method file
--servers myLDAPserver --base-dn dc=example,dc=com
--user-name cn=manager,dc=example,dc=com
--netbios-name ess --enable-server-tls
```

The system displays the following output:

```
File authentication configuration completed successfully.
```

3. Issue the **mmuserauth service list** command to see the current authentication configuration as shown
   in the following example:

   ```
   # mmuserauth service list
   ```

   The system displays the following output:

   ```
   FILE access configuration : LDAP
   PARAMETERS              VALUES
   -------------------------------------------------
   ENABLE_SERVER_TLS       true
   ENABLE_KERBEROS         false
   USER_NAME               cn=manager,dc=example,dc=com
   SERVERS                 myLDAPserver
   NETBIOS_NAME            ess
   BASE_DN                 dc=example,dc=com
   USER_DN                 none
   GROUP_DN                none
   NETGROUP_DN             none
   USER_OBJECTCLASS        posixAccount
   GROUP_OBJECTCLASS       posixGroup
   USER_NAME_ATTRIB        cn
   USER_ID_ATTRIB          uid
   KERBEROS_SERVER         none
   KERBEROS_REALM          none

   OBJECT access not configured
   PARAMETERS              VALUES
   -------------------------------------------------
   ```

4. Verify the user resolution on system present in LDAP:

   ```
   # id ldapuser2
   uid=1001(ldapuser2) gid=1001(ldapuser2) groups=1001(ldapuser2)
   ```

Configuring LDAP with Kerberos for file access

You can configure LDAP with Kerberos as the authentication method for file access. Using Kerberos with
LDAP provides more security for the communication channel between the IBM Spectrum Scale system
and LDAP server.

Example for configuring LDAP with Kerberos as the authentication method for file access.

1. Ensure that the keytab file is also placed under the /var/mmfs/tmp directory with the name as
   krb5.keytab on the node where the command is run. Perform validation of keytab file availability
   with desired name at required location:

   ```
   # stat /var/mmfs/tmp/krb5.keytab
   File: /var/mmfs/tmp/krb5.keytab
   Size: 502 Blocks: 8 IO Block: 4096 regular file
   Device: fd00h/64768d Inode: 103169898 Links: 1
   Access: (0600/-rw-------) Uid: ( 0/ root) Gid: ( 0/ root)
   Context: unconfined_u:object_r:user_tmp_t:s0
   Access: 2015-01-23 14:31:18.244837381 +0530
   Modify: 2015-01-23 12:45:05.475837381 +0530
   Change: 2015-01-23 12:45:05.476837381 +0530
   Birth: -
   ```

2. Issue the **mmuserauth service create** command as shown in the following example:

   ```
   # mmuserauth service create --type ldap --data-access-method file
   --servers myLDAPserver --base-dn dc=example,dc=com
   --user-name cn=manager,dc=example,dc=com
   --netbios-name ess --enable-kerberos --kerberos-server myKerberosServer
   --kerberos-realm example.com
   ```

   The system displays the following output:

   ```
   File authentication configuration completed successfully.
   ```

3. Issue the **mmuserauth service list** command to see the current authentication configuration as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_SERVER_TLS       false
ENABLE_KERBEROS         true
USER_NAME               cn=manager,dc=example,dc=com
SERVERS                 myLDAPserver
NETBIOS_NAME            ess
BASE_DN                 dc=example,dc=com
USER_DN                 none
GROUP_DN                none
NETGROUP_DN             none
USER_OBJECTCLASS        posixAccount
GROUP_OBJECTCLASS       posixGroup
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
KERBEROS_SERVER         myKerberosServer
KERBEROS_REALM          example.com

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

Configuring LDAP with TLS and Kerberos for file access

You can configure LDAP with TLS and Kerberos as the authentication method for file access. Using Kerberos and TLS with LDAP provides maximum security for the communication channel between the IBM Spectrum Scale system and the LDAP server.

Provides an example on how to configure LDAP with TLS and Kerberos as the authentication method for file access.

1. Ensure that the CA certificate for LDAP server is placed under /var/mmfs/tmp directory with the name ldap_cacert.pem; specifically, on the protocol node where the command is run. Perform validation of CA cert availability with desired name at the required location as shown in the following example:

```
# stat /var/mmfs/tmp/ldap_cacert.pem
File: /var/mmfs/tmp/ldap_cacert.pem
Size: 2130 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 12:37:34.088837381 +0530
Modify: 2015-01-23 12:16:24.438837381 +0530
Change: 2015-01-23 12:16:24.438837381 +0530
```

2. Ensure that the keytab file is placed under /var/mmfs/tmp directory name as krb5.keytab specifically on the node where the command is run. Perform validation of keytab file availability with desired name at the required location:

```
# stat /var/mmfs/tmp/krb5.keytab
File: /var/mmfs/tmp/krb5.keytab
Size: 502 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169898 Links: 1
Access: (0600/-rw-------) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 14:31:18.244837381 +0530
Modify: 2015-01-23 12:45:05.475837381 +0530
Change: 2015-01-23 12:45:05.476837381 +0530
Birth: -
```

3. Issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method file
--servers myLDAPserver --base-dn dc=example,dc=com
--user-name cn=manager,dc=example,dc=com
--netbios-name ess --enable-server-tls --enable-kerberos
--kerberos-server myKerberosServer --kerberos-realm example.com
```

The system displays the following output:

```
File authentication configuration completed successfully.
```

4. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_SERVER_TLS       true
ENABLE_KERBEROS         true
USER_NAME               cn=manager,dc=example,dc=com
SERVERS                 myLDAPserver
NETBIOS_NAME            ess
BASE_DN                 dc=example,dc=com
USER_DN                 none
GROUP_DN                none
NETGROUP_DN             none
USER_OBJECTCLASS        posixAccount
GROUP_OBJECTCLASS       posixGroup
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
KERBEROS_SERVER         myKerberosServer
KERBEROS_REALM          example.com

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

5. Verify the user resolution on the system:

```
# id ldapuser3
uid=1002(ldapuser3) gid=1002(ldapuser3) groups=1002(ldapuser3)
```

Configuring LDAP without TLS and Kerberos for file access

You can configure LDAP without TLS or Kerberos for file access. But this method is less secured compared to LDAP with TLS, LDAP with TLS and Kerberos, and LDAP with Kerberos configurations.

Provides an example on how to configure LDAP without TLS and Kerberos as the authentication method for file access.

1. Issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method file
--servers 192.0.2.18 --base-dn dc=example,dc=com
--user-name cn=manager,dc=example,dc=com  --netbios-name ess
```

The system displays the following output:

```
File Authentication configuration completed successfully.
```

2. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_SERVER_TLS       false
ENABLE_KERBEROS         false
USER_NAME               cn=manager,dc=example,dc=com
SERVERS                 192.0.2.18
NETBIOS_NAME            ess
BASE_DN                 dc=example,dc=com
USER_DN                 none
GROUP_DN                none
NETGROUP_DN             none
USER_OBJECTCLASS        posixAccount
GROUP_OBJECTCLASS       posixGroup
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
KERBEROS_SERVER         none
KERBEROS_REALM          none

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

Configuring NIS-based authentication

The Network Information Service (NIS)-based authentication is useful in NFS-only environment where NIS acts as an ID mapping server and also used for netgroups. When file access is configured with NIS, SMB access cannot be enabled.

Ensure that you have the following details before you start NIS-based authentication:
- NIS domain name. This is case-specific
- IP address or host name of the NIS server
- Primary DNS is added in the /etc/resolv.conf file on all the protocol nodes. It resolves the authentication server system with which the IBM Spectrum Scale system is configured. The manual changes done to the configuration files might get overwritten by the Operating System's network manager. So, ensure that the DNS configuration is persistent even after you restart the system. For more information on the circumstances where the configuration files are overwritten, refer the corresponding Operating System documentation.

You need to run the **mmuserauth service create** command with the following mandatory parameters to configure NIS as the authentication method:
- --type nis
- --data-access-method file
- --domain domainName
- --servers comma-delimited IP address or host name

For more information on each parameter, see the **mmuserauth service create** command.

Provides an example on how to configure NIS as the authentication method for file access.
1. Issue the **mmuserauth service create** command as shown in the following example:

   ```
   # mmuserauth service create --type nis --data-access-method file
   --servers myNISserver --domain nisdomain3
   ```

   The system displays the following output:

   ```
   File Authentication configuration completed successfully.
   ```
2. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

   ```
   # mmuserauth service list
   ```

The system displays the following output:

```
FILE access configuration : NIS
PARAMETERS               VALUES
-------------------------------------------------
SERVERS                  myNISserver
DOMAIN                   nisdomain3

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------.
```

Authentication considerations for NFSv4 based access

This topic describes authentication considerations for NFSv4 based access.

NFSv4 user name mapping configuration

1. To enable NFSv4 access, NFS server user name mapping configuration on IBM Spectrum Scale should be updated. Issue the following command:

   `mmnfs config change "IDMAPD_DOMAIN=myDomain.com"`

2. On the NFS client, ID map configuration should also be updated to reflect the same domain name as defined on NFS server. Additionally ID mapping service should be started.

   For example, on RHEL 7.x NFS clients

   - The ID map configuration file name is `etc/idmapd.conf`. Update the Domain attribute in the file to reflect the domain name defined on the NFS server.

   - Start `nfs.idmap service`.

   Note: The ID map configuration file and ID mapping service can differ on various OS platforms.

Prerequisites for configuring Kerberos based NFS access

This topic describes the requirement that must be met to configure IBM Spectrum Scale for Kerberized NFS access.

General requirements

- For Kerberized NFS access, time must be synchronized across the KDC server, the IBM Spectrum Scale cluster protocol nodes, and the NFS clients. Otherwise, access to an NFS export might be denied.

- For Kerberized NFSv3 access, NFS clients should mount NFS exports by using one of the configured CES IP addresses.

- For Kerberized NFSv4 access, NFS clients can mount NFS exports by using either "one of the configured CES IP addresses" or the "system account name" that is configured for FILE protocols authentication. The "system account name" is the value that is specified for the --netbios-name option in the **mmuserauth CLI** command during FILE protocols authentication configuration.

IBM Spectrum Scale NFS server configuration for Kerberos access

- To enable NFS Kerberos access, update the NFS server configuration with the Kerberos realm name. Issue the following command to configure NFS configuration parameter LOCAL_REALMS:

   `mmnfs config change "LOCAL_REALMS=MYREALM.COM"`

   Set this attribute to the KDC REALM value.

   Note: Specify the realm name in capital letters.

- Configure the same local realms value (for example, MYREALM.COM here) on all NFS Kerberos clients (for example, on RHEL NFS clients set Local-Realms attribute in the `/etc/idmapd.conf` file). This configuration file might be different on various client OS systems.

- On NFS client, ID map configuration should also be updated to reflect the same realm name as defined on NFS server. Additionally, the service for establishing Kerberos access with NFS server should also be started. For example, on RHEL 7.X NFS clients, the ID map configuration file name is `etc/idmapd.conf`. Update the Local-Realms attribute in the file to reflect the Kerberos realm defined on NFS server and then start the nfs.secure service.

  Note: The ID map configuration file and service to establish secure access can differ on various OS platforms.

Considerations for LDAP based authentication schemes

- In LDAP based authentication schemes, administrators must generate keytab file prior to FILE protocols authentication configuration. The keytab file should be generated on the KDC server and then copied to path `/var/mmfs/tmp/` on the IBM Spectrum Scale node. The **mmuserauth** command must be initiated from the node where the keytab file is copied.
- The keytab file should contain NFS service principals of short name and FQDN of the "system account name". The service principal name format is `nfs/<system account name>@<KERBEROS REALM>`. For example, If the "system account name" is FOO, "system account FQDN" is FOO.MYDOMAIN.COM and the "realm" is MYREALM.COM, then service principals required to be created should be `nfs/FOO@MYREALM.COM` and `nfs/FOO.MYDOMAIN.COM@MYREALM.COM`.
- The realm name is the value specified for the --kerberos-realm-option in the **mmuserauth** command.

Considerations for AD based authentication schemes

- In Active Directory based authentication schemes, administrators need not prepare a keytab file. The **mmuserauth CLI** command prepares keytab file during FILE protocols authentication configuration. It adds NFS service principals of short name and FQDN for "system account name" in the local keytab file placed at `/etc/krb5.keytab` on all the protocol nodes in the CES cluster.
- User must specify the --enable-nfs-kerberos option in the **mmuserauth** command to activate the NFS kerberized access to IBM Spectrum Scale.

Managing user-defined authentication

In the user-defined mode of authentication, the user is free to select the authentication and ID mapping methods of their choice. It is the responsibility of the administrator of the client system to manage the authentication and ID mapping for file (NFS and SMB) and object access to the IBM Spectrum Scale system.

The IBM Spectrum Scale system administrators are not allowed use any of the GPFS commands to manage authentication. It is important for the end user to be aware of the limitations, if any, of the authentication and ID mapping scheme that will be implemented after configuring the user-defined mode of authentication.

The user-defined mode is appropriate in the following circumstances:

- The client already has protocol deployments either on GPFS installations or on different systems and is planning to move to using the protocol stack on the IBM Spectrum Scale system. The client wants to replicate the current authentication and ID mapping configuration. In this case, the client system administrator must be familiar with the required configuration settings that will be applied to the system.
- If the end user wants an authentication method that is not supported by the IBM Spectrum Scale system.

Note: If the end user wants to configure the authentication methods that are supported by the IBM Spectrum Scale system, it is highly recommended to configure the authentication and ID mapping methods by using the **mmuserauth** command instead of opting for the user-defined method of authentication.

The IBM Spectrum Scale system administrator needs to specify that the user-defined mode of authentication is used by using the **--type userdefined** option in the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type userdefined --data-access-method file
File Authentication configuration completed successfully.
```

Submit the **mmuserauth service list** command to see the current authentication configuration as shown in the following example:

```
# mmuserauth service list
FILE access configuration : USERDEFINED
PARAMETERS                VALUES
-------------------------------------------------
OBJECT access not configured
PARAMETERS                VALUES
-------------------------------------------------
```

Typically, user-defined authentication is used when existing GPFS customers are already using GPFS with NFS and do not want to alter the authentication that is already configured on these systems. You can configure user-defined authentication for both object and file access or for object or file alone.

Note: Authorization depends upon authentication and ID mapping that is configured with the system. That is, the ACL control on exports, files, and directories depend on the authentication method that is configured.

| **File authentication configuration**

Ensure the following while using the user-defined mode of authentication for file access:

- Ensure that the authentication server and ID mapping server are always reachable from all the protocol nodes. For example, if NIS is configured as the ID mapping server, you can use the 'ypwhich' command to ensure that NIS is configured and reachable from all the protocol nodes. Similarly, if LDAP is configured as authentication and ID mapping server, you can bind to the LDAP server from all protocol nodes to monitor if the LDAP server is reachable from all protocol nodes.
- Ensure that the implemented authentication and ID mapping configuration is always consistent across all the protocol nodes. This requires that the authentication server and ID mapping server are manually maintained and monitored by the administrator. The administrator must also ensure that the configuration files are not overwritten due to node restart and other similar events.
- Ensure that the implemented authentication and ID mapping-related daemons and processes across the protocol nodes are always up and running.
- The users or groups, accessing the IBM Spectrum Scale system over NFS and SMB protocols must resolve to a unique UID and GID respectively on all protocol nodes, especially in implementations where different servers are used for authentication and ID mapping. The name that is registered in ID mapping server for user and group must be checked for resolution.

  For example:

  ```
  # id fileuser
  uid=1234(fileuser) gid=5678(filegroup) groups=5678(filegroup)
  ```

  Note: However, there are some use cases where only NFSV3 based access to the IBM Spectrum Scale system is used. In such cases, the user and group IDs are obtained from the NFS client and there is no ID mapping setting is configured on the protocol nodes.
- If the IBM Spectrum Scale system is configured for multiprotocol support (that is, the same data is accessed through both NFS and SMB protocols), ensure that the IDs of users and groups are consistent across the NFS clients and SMB clients and that they resolve uniquely on the protocol nodes.
- Ensure that there is no conflict of UID and GID across users and groups that are accessing the system. This must be strictly enforced, especially in multiprotocol-based access deployments.

- Ensure that the Kerberos configuration files, placed on all protocol nodes, are in synchronization with each other. Ensure that the clients and the IBM Spectrum Scale system are part of the same Kerberos realm or trusted realm.
- While deploying two or more IBM Spectrum Scale clusters, ensure that the ID mapping is consistent in cases where you want to use IBM Spectrum Scale features like AFM, AFM-DR, and asynchronous replication of data.

The following table provides an overview of the authentication requirements for each file access protocol. Refer this table when you plan to use user-defined mode as the authentication method.

Table 19. Authentication requirements for each file access protocol.

File access protocol	Requirements
NFSV3	In scenarios where user name and group name are expected to be used to native GPFS commands (for example, setting data ownership, listing user or group quota), the IBM Spectrum Scale system must be able to resolve the UID and GID to user name and group name and vice versa, consistently across all the protocol nodes. **Note:** However, there are some use cases where only the NFSv3 based access to the IBM Spectrum Scale system is used. In such cases, the user and group IDs are coming from the NFS client and there is no ID mapping setting is configured on the protocol nodes.
Kerberos NFSV3	Ensure that the user name and group name that are used to access data consistently resolve to same UID and GID across all protocol nodes and NFS clients. Ensure that the time is synchronized on the NFS server, NFS clients, and Kerberos server. **Note:** User names and group names are case-sensitive.
NFSV4	Ensure that the user name and group name that are used to access data consistently resolve to same UID and GID across all protocol nodes and NFS clients. Domain name must be specified in the `/etc/idmapd.conf` file and it must be the same on both the NFS server and NFS clients. **Note:** User names and group names are case-sensitive.
Kerberos NFS V4	Ensure that the user name and group name that are used to access data consistently resolve to same UID and GID across all protocol nodes and NFS clients. Ensure that the time is synchronized on the NFS server, NFS clients, and Kerberos server. Domain name and local-realms must be specified in the `/etc/idmapd.conf` file and it must be the same on both the NFS server and NFS clients. The value of "local-realms" takes the value of Kerberos realm with which the IBM Spectrum Scale system protocol nodes are configured.

Table 19. Authentication requirements for each file access protocol. (continued)

File access protocol	Requirements
SMB	Ensure that the user name and group name that are used to access data consistently resolve to same UID and GID across all protocol nodes and NFS clients. While integrating with non-windows server, ensure that the samba attributes are populated on the directory server for every user and group that are planning to access the IBM Spectrum Scale system. Special care must be taken to match the samba domain SIDs. For Kerberized SMB access, ensure that time is synchronized the SMB server, SMB client, and Kerberos server.

Object authentication configuration

The user-defined mode for object authentication integrates IBM Spectrum Scale Object Storage with the externally hosted keystone server. Ensure the following while using the user-defined mode of authentication for object access:

- Integration with external keystone server is supported over http and https.
- The specified object user must be defined while enabling and configuring object in the external keystone server.
- The 'service' tenant/project must be defined in the external keystone server.
- The 'admin' role must be defined in the external keystone server.
- Ensure that the specified swift user has 'admin' role in 'service' tenant/project.

  For example, the external keystone server must contain the following admin role definition to sift user:

  ```
  # openstack rol list --user swift --project service
  +----------------------------------+-------+---------+-------+
  | ID | Name | Project | User |
  +----------------------------------+-------+---------+-------+
  | 90877d1913964e1eac05031e45afb46a | admin | service | swift |
  +----------------------------------+-------+---------+-------+
  ```

- The users and projects must be mapped to the `Default` domain in Keystone.
- Object storage service endpoints must be correctly defined in the external keystone server.

  For example, the external keystone server must contain the following endpoint for object-store:

  ```
  # openstack endpoint list
  +------------+--------+--------------+--------------+---------+-----------+------------------------------------------------------------------+
  | ID         | Region | Service Name | Service Type | Enabled | Interface | URL                                                              |
  +------------+--------+--------------+--------------+---------+-----------+------------------------------------------------------------------+
  | c36e..9da5 | None   | keystone     | identity     | True    | public    | http://specscaleswift.example.com:5000/                          |
  | f4d6..b040 | None   | keystone     | identity     | True    | internal  | http://specscaleswift.example.com:35357/                         |
  | d390..0bf6 | None   | keystone     | identity     | True    | admin     | http://specscaleswift.example.com:35357/                         |
  | 2e63..f023 | None   | swift        | object-store | True    | public    | http://specscaleswift.example.com:8080/v1/AUTH_%(tenant_id)s     |
  | cd37..9597 | None   | swift        | object-store | True    | internal  | http://specscaleswift.example.com:8080/v1/AUTH_%(tenant_id)s     |
  | a349..58ef | None   | swift        | object-store | True    | admin     | http://specscaleswift.example.com:8080                           |
  +------------+--------+--------------+--------------+---------+-----------+------------------------------------------------------------------+
  ```

If the object authentication is set to 'user-defined' and an IP address/port number is set in the proxy server configuration for keystone authentication, then that IP address will be checked using a simple http(s) request. If the request fails, then the AUTH_OBJ state will be set to "degraded" and an 'external keystone URL failure' event will be logged. This will not cause the node to be flagged as bad nor will it cause any public IP movement.

- Issue the following command:

  ```
  mmces state show
  ```

The system displays output similar to this:

```
NODE                     AUTH      AUTH_OBJ  NETWORK NFS       OBJ       SMB      CES
spectrum-31.localnet.com DISABLED  DEGRADED  HEALTHY DISABLED  DEGRADED  HEALTHY  DEGRADED
```

- Issue the following command:

```
mmces events list
```

The system displays output similar to this:

```
NODE                     TIMESTAMP                            EVENT NAME    SEVERITY  DETAILS
spectrum-31.localnet.com 2015-10-18 18:23:05.386336--1:-1CEST ks_url_exfail WARNING   Keystone request failed
                                                              using http://10.11.0.1:35357/v2.0
```

Configuring authentication for object access

Configuring authentication for object access by using the command line utility

There are two methods of configuring authentication:

1. By using the installation toolkit
2. By using the **mmuserauth** command in the command line utility

You can use the following authentication methods for object access:

- Active Directory (AD)
- LDAP
- Local authentication
- User-defined (external keystone)

The AD-based and LDAP-based authentication methods use an external AD and LDAP server respectively to manage the authentication. Local authentication is handled by a Keystone server that resides within the IBM Spectrum Scale system.

The IBM Spectrum Scale system installation process configures Keystone server that is required for object access. By default the IBM Spectrum Scale installation process configures object authentication with a local Keystone authentication method. If you have an existing Keystone server that you want to use, specify that it be used for authentication.

Before you configure object authentication method, ensure that the Keystone Identity service is properly configured.

Note: Before you configure an authentication method for object access, ensure that all protocol nodes have CES IP addresses assigned and you are issuing the authentication configuration command from the protocol node that has one or more CES IP addresses assigned to it.

Before you start manually configuring authentication method for object access, ensure that the `openldap-clients` RPM is installed.

On each protocol node, issue the following command: **yum install openldap-clients**.

Note: This step is required only when the authentication type is AD/LDAP.

The mapping between user, role, and tenant is stored in the Keystone database. If you switch from one authentication type to another you must delete the existing mapping definitions by running the **mmuserauth service remove --idmapdelete** command.

Note:

It is recommended to run the **mmuserauth service check** command as follows after configuring object authentication using the **mmuserauth service create** command:

```
mmuserauth service check --data-access-method object -N cesNodes
```

If the **mmuserauth service check** command reports that any certificate file is missing on any of the nodes, then run the following command:

```
mmuserauth service check --data-access-method object -N cesNodes --rectify
```

For more information about **mmuserauth service check**, see the topic *mmuserauth command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Configuring local authentication for object access

Object access can be configured with the Keystone server that is available in the IBM Spectrum Scale system. In this mode, Keystone stores the identity and assignment information locally in its database.

The local authentication method is useful when you want to create and maintain a separate set of users for only object access. These users cannot use the local authentication credentials for accessing file data that is hosted through NFS and SMB protocols. If you want to allow a user to access both file and object, use an external authentication server such as AD or LDAP to manage user accounts and authentication requests.

Note: File and object authentication must be configured with individual invocations of the **mmuserauth** command, even if the authentication server is the same.

You need to use the **mmuserauth service create** command with the following mandatory parameters to configure local authentication for object access:
- --type local
- --data-access-method `object`
- --ks-admin-user `keystoneAdminName`

For more information on each parameter, see the **mmuserauth service create** command.

1. To configure local authentication for object access, issue **mmuserauth service create** command as shown in the following example:

   ```
   # mmuserauth service create --data-access-method object --type local
   --ks-dns-name c40bbc2xn3 --ks-admin-user admin
   ```

   The system displays the following output:

   ```
   Object configuration with local (Database) as identity backend is completed
   successfully.
   Object Authentication configuration completed successfully.
   ```

2. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

   ```
   # mmuserauth service list
   ```

   The system displays the following output:

   ```
   FILE access not configured
   PARAMETERS              VALUES
   -------------------------------------------------

   OBJECT access configuration : LOCAL
   PARAMETERS              VALUES
   -------------------------------------------------
   ENABLE_KS_SSL           false
   ENABLE_KS_CASIGNING     false
   KS_ADMIN_USER           admin
   ```

Configuring local authentication with SSL for object access

Use the following steps to configure object access with the Keystone server that is available in the IBM Spectrum Scale system with SSL enabled.

1. Obtain certificates from the Certification Authority (CA) and place them at the following location on the current node from where the **mmuserauth service create** command is being executed.

   ```
   /var/mmfs/tmp/ssl_cert.pem
   /var/mmfs/tmp/ssl_key.pem
   /var/mmfs/tmp/ssl_cacert.pem
   ```

   Note:
   - Self-signed certificates can be used for testing and demonstration purposes. However, the use of externally signed certificates is strongly recommended for production environments.
   - The name in the SSL certificate must match the Keystone endpoint name.

2. Remove existing local authentication for object access as follows.

   ```
   mmuserauth service remove --data-access-method object
   ```

3. Configure local authentication with SSL for object access as follows.

   ```
   mmuserauth service create --data-access-method object --type local --enable-ks-ssl
   ```

Local authentication is now configured for object access with SSL enabled.

To disable SSL and configure local authentication for object access again, use the following steps.

4. Remove existing local authentication for object access as follows.

   ```
   mmuserauth service remove --data-access-method object
   ```

   If you are also changing authentication type, remove authentication and ID mappings by using the following commands in sequence.

   ```
   mmuserauth service remove --data-access-method object
   mmuserauth service remove --data-access-method object --idmapdelete
   ```

5. Configure local authentication without SSL for object access as follows.

   ```
   mmuserauth service create --data-access-method object --type local
   ```

Configuring an AD-based authentication for object access

You can configure Keystone with an external AD server as the authentication back-end so that AD users can access the object store by using their AD credentials. The same AD server can be used for both object access and file access.

The AD server is set up to handle the authentication requests. AD is used as an LDAP server. Unlike file access, multiple AD domains are not supported.

Prerequisites

Ensure that you have the following details before you start AD-based authentication configuration:
- AD server details such as IP address or host name, user name, user password, base dn, and user dn.
- If you want to configure TLS with AD for secure communication between Keystone and AD, you need to place the CA certificate that is used for signing the AD server setup for TLS under the following directory of the node on which the **mmuserauth service create** command is run:
  - `/var/mmfs/tmp/ldap_cacert.pem`
- The secret key you provided for encrypting/decrypting passwords unless you have disabled prompting for the key.

See "Integrating with AD server" on page 175 for more information on the prerequisites for integrating AD server with the IBM Spectrum Scale system.

The following parameters must be used with **mmuserauth service create** command to configure AD-based authentication for object access:

- **--type** ad
- **--data-access-method** object
- **--servers** IP address or host name of AD. All user lookups by Keystone are done only against this server. If multiple servers are specified, only the first server is used and the rest are ignored.
- **--base-dn** ldapBase
- { **--enable-anonymous-bind** | **--user-name BindDN --password BindPwd**} You need to mention either anonymous bind or either –user-name or –password.
- **--enable-server-tls**, if TLS needs to be enabled
- **--user-dn** ldapUserSuffix. LDAP container from where users are looked up.
- **--ks-admin-user** keystoneAdminUser from AD
- **--enable-ks-ssl**, if SSL needs to be enabled. You need to have another set of certificates that are placed in standard directory.
- **--enable-ks-casigning**, if you want to use external CA signed certificate for token signing
- **--ks-swift-user** Swift_Service_User from AD
- **--ks-swift-pwd** Swift_Service_User's Password from AD

For more information on each parameter, see the **mmuserauth service create** command.

To change the authentication method that is already configured for object access, you need to remove the authentication method and ID mappings. For more information, see "Deleting the authentication and the ID mapping configuration" on page 215.

Configuring AD without TLS for object access

Configuring AD without TLS does not provide secured communication between the IBM Spectrum Scale system and the authentication server.

1. Submit the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ad --data-access-method object
--user-name "cn=Administrator,cn=Users,dc=IBM,dc=local"
--base-dn "dc=IBM,DC=local" --ks-dns-name c40bbc2xn3 --ks-admin-user admin
--servers myADserver --user-id-attrib cn --user-name-attrib sAMAccountName
--user-objectclass organizationalPerson --user-dn "cn=Users,dc=IBM,dc=local"
--ks-swift-user swift
```

   The system displays the following output:

```
Object configuration with LDAP (Active Directory) as identity
backend is completed successfully.
Object Authentication configuration completed successfully.
```

2. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

   The system displays the following output:

```
FILE access not configured
PARAMETERS              VALUES
-------------------------------------------------

OBJECT access configuration: AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       false
ENABLE_KS_SSL           false
USER_NAME               cn=Administrator,cn=Users,dc=IBM,dc=local
```

```
SERVERS              myADserver
BASE_DN              dc=IBM,DC=local
USER_DN              cn=users,dc=ibm,dc=local
USER_OBJECTCLASS     organizationalPerson
USER_NAME_ATTRIB     sAMAccountName
USER_ID_ATTRIB       cn
USER_MAIL_ATTRIB     mail
USER_FILTER          none
ENABLE_KS_CASIGNING  false
KS_ADMIN_USER        admin
```

Configuring AD with TLS for object access

Configuring AD with TLS helps to encrypt the communication between the IBM Spectrum Scale system and AD server.

Configures AD with TLS as the authentication method for object access.

1. Ensure that the CA certificate for AD server is placed under /var/mmfs/tmp directory with the name ldap_cacert.pem specifically on the protocol node where the command is run. Perform validation of CA cert availability with desired name at required location as shown in the following example:

```
# stat /var/mmfs/tmp/ldap_cacert.pem
File: /var/mmfs/tmp/ldap_cacert.pem
Size: 2130 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 12:37:34.088837381 +0530
Modify: 2015-01-23 12:16:24.438837381 +0530
Change: 2015-01-23 12:16:24.438837381 +0530
```

2. To configure AD with TLS authentication for object access, issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ad --data-access-method object
--user-name "cn=Administrator,cn=Users,dc=IBM,dc=local"
--base-dn "dc=IBM,DC=local" --enable-server-tls --ks-dns-name c40bbc2xn3
--ks-admin-user admin --servers myADserver --user-id-attrib cn
--user-name-attrib sAMAccountName --user-objectclass organizationalPerson
--user-dn "cn=Users,dc=IBM,dc=local" --ks-swift-user swift
```

The system displays the following output:

```
Object configuration with LDAP (Active Directory) as identity
backend is completed successfully.
Object Authentication configuration completed successfully.
```

3. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access not configured
PARAMETERS           VALUES
-------------------------------------------------

OBJECT access configuration: AD
PARAMETERS           VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND  false
ENABLE_SERVER_TLS      true
ENABLE_KS_SSL          false
USER_NAME              cn=Administrator,cn=Users,dc=IBM,dc=local
SERVERS                myADserver
BASE_DN                dc=IBM,DC=local
USER_DN                cn=users,dc=ibm,dc=local
USER_OBJECTCLASS       organizationalPerson
```

```
USER_NAME_ATTRIB          sAMAccountName
USER_ID_ATTRIB            cn
USER_MAIL_ATTRIB          mail
USER_FILTER               none
ENABLE_KS_CASIGNING       false
KS_ADMIN_USER             admin
```

Configuring an LDAP-based authentication for object access

You can configure Keystone with an external LDAP server as the authentication back-end. This will allow LDAP users to access the object store using their LDAP credentials. The same LDAP server can be used for both object access and file access.

Prerequisites

Ensure that you have the following details before you configure LDAP-based authentication:

- LDAP server details such as IP address or host name, LDAP user name, user password, base dn, and user dn.
- If you want to configure TLS with LDAP for secure communication between Keystone and LDAP, you need to place the CA certificate that is used for signing the LDAP server setup for TLS under the following directory of the node on which the **mmuserauth service create** command is run:
  - /var/mmfs/tmp/ldap_cacert.pem
- The secret key you provided for encrypting/decrypting passwords unless you have disabled prompting for the key.

See "Integrating with LDAP server" on page 176 for more information on the prerequisites for integrating LDAP server with the IBM Spectrum Scale system.

You need to issue the **mmuserauth service create** command to configure LDAP-based authentication with the following parameters:

- **--type** ldap
- **--data-access-method** object
- **--servers** IP address or host name of LDAP (all user lookups by Keystone is done only against this server. If multiple servers are specified, only the first server is used and rest are ignored).
- **--base-dn** ldapBase
- { **--enable-anonymous-bind** | **--user-name BindDN --password BindPwd**} (You need to mention either anonymous bind or either **--user-name** or **--password**).
- **--enable-server-tls**, if TLS needs to be enabled.
- **--user-dn** ldapUserSuffix (LDAP container from where users are looked up)
- **--ks-admin-user** keystoneAdminUser from LDAP.
- **--enable-ks-ssl**, if SSL needs to be enabled. You need to have another set of certificates that are placed in the standard directory.
- **--enable-ks-casigning**, if you want to use external CA signed certificate for token signing.
- **--ks-swift-user** swiftServiceUser from LDAP.
- **--ks-swift-pwd** swiftServiceUser Password from LDAP.

For more information on each parameter, see the **mmuserauth service create** command.

To change the authentication method that is already configured for object access, you need to remove the authentication method and ID mappings. For more information, see "Deleting the authentication and the ID mapping configuration" on page 215.

Configuring LDAP without TLS for object access

Perform the following steps to configure LDAP-based authentication for object access:

1. To configure LDAP-based authentication for object access, issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method object
--user-name "cn=manager,dc=essldapdomain"
--base-dn dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com --ks-dns-name c40bbc2xn3
--ks-admin-user mamdouh --servers 192.0.2.11
--user-dn "ou=People,dc=essldapdomain" --ks-swift-user swift
```

   The system displays the following output:

```
Object configuration with LDAP as identity backend is completed successfully.
Object Authentication configuration completed successfully.
```

2. To verify the authentication configuration, issue the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

   The system displays the following output:

```
FILE access not configured
PARAMETERS              VALUES
-------------------------------------------------

OBJECT access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       false
ENABLE_KS_SSL           false
USER_NAME               cn=manager,dc=essldapdomain
SERVERS                 192.0.2.11
BASE_DN                 dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com
USER_DN                 ou=people,dc=essldapdomain
USER_OBJECTCLASS        posixAccount
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
USER_MAIL_ATTRIB        mail
USER_FILTER             none
ENABLE_KS_CASIGNING     false
KS_ADMIN_USER           mamdouh
```

Configuring LDAP with TLS for object access

Perform the following steps to configure LDAP with TLS-based authentication for object access:

1. Ensure that the CA certificate for the LDAP server is placed under /var/mmfs/tmp directory with the name ldap_cacert.pem specifically on the protocol node where the command is run. Perform validation of CA cert availability with desired name at required location as shown in the following example:

```
# stat /var/mmfs/tmp/ldap_cacert.pem
File: /var/mmfs/tmp/ldap_cacert.pem
Size: 2130 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 12:37:34.088837381 +0530
Modify: 2015-01-23 12:16:24.438837381 +0530
Change: 2015-01-23 12:16:24.438837381 +0530
```

2. To configure LDAP with TLS-based authentication for object access, issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method object
--user-name "cn=manager,dc=essldapdomain"
--base-dn dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com --enable-server-tls
--ks-dns-name c40bbc2xn3 --ks-admin-user mamdouh --servers 192.0.2.11
--user-dn "ou=People,dc=essldapdomain" --ks-swift-user swift
```

The system displays the following output:

```
Object configuration with LDAP as identity backend is completed successfully.
Object Authentication configuration completed successfully.
```

3. To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access not configured
PARAMETERS              VALUES
-------------------------------------------------

OBJECT access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       true
ENABLE_KS_SSL           false
USER_NAME               cn=manager,dc=essldapdomain
SERVERS                 192.0.2.11
BASE_DN                 dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com
USER_DN                 ou=people,dc=essldapdomain
USER_OBJECTCLASS        posixAccount
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
USER_MAIL_ATTRIB        mail
USER_FILTER             none
ENABLE_KS_CASIGNING     false
KS_ADMIN_USER           mamdouh
```

Configuring object authentication with an external keystone server

The object protocol can be configured with an external keystone server. This can be accomplished by either using an existing internal keystone server that is already deployed in the local environment or by utilizing an external keystone server that is hosted outside of the local environment.

The following prerequisites must be met before you start configuring an external keystone server with the IBM Spectrum Scale system.

- The external keystone server must be running and reachable from all protocol nodes.
- The keystone server administrator must create an object storage service for the required user, for object authentication configuration.

To configure an external keystone server with the IBM Spectrum Scale system, enter the **mmuserauth service create** command as shown in the following example:

-
  ```
  mmuserauth service create --data-access-method object --type userdefined
  --ks-ext-endpoint http://specscaleswift.example.com:35357/v3
  --ks-swift-user swift
  ```

Configuring IBM Spectrum Scale for object storage with SSL-enabled external keystone

1. Remove the object authentication along with the ID mapping ID if it is present by running one of the following commands:

```
mmuserauth service remove --data-access-method object
mmuserauth service remove --data-access-method object --idmapdelete
```

2. Copy the CA certificate with the external keystone to the node where the **mmuserauth** command is being run in directory /var/mmfs/tmp, for example:

   ```
   /var/mmfs/tmp/ks_ext_cacert.pem
   ```

3. Configure the object authentication by running the **mmuserauth service create** command with the `--enable-ks-ssl` option:

   ```
   mmuserauth service create --data-access-method object --type userdefined
   --ks-ext-endpoint https://specscaleswift.example.com:35357/v3
   --ks-swift-user swift  --enable-ks-ssl
   ```

Note: Object configuration with SSL-enabled external keystone is not supported on the installer toolkit and **mmobj swift base**.

Creating object accounts

An account is used to group or isolate object resources. Each object user is part of an account. Object users are mapped to an account and can access only the objects that reside within the project. Each user needs to be defined with a set of user rights and privileges to perform a specific set of operations on the resources of the account to which it belongs. Users can be assigned to multiple accounts with different roles on each account.

You must create at least one account before adding users. An account contains a list of containers in the object storage. You can also define quota at the account level. An object account represents a storage location for a project rather than a specific user.

Note:
To work with this function in the IBM Spectrum Scale GUI, log on to the GUI and select **Object** > **Accounts**.

1. To view the details for an existing account, issue the **swift stat** command:

   ```
   swift stat --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3 \
               --os-project-name admin \
               --os-project-domain-name Default \
               --os-username admin \
               --os-user-domain-name Default \
               --os-password Passw0rd \
               --auth-version 3
   ```

   or

   ```
   source openrc
   swift stat
   ```

   The system displays output similar to the following:

   ```
   Account: AUTH_bea5a0c632e54eaf85e9150a16c443cet
   Containers: 0
   Objects: 0
   Bytes: 0
   X-Put-Timestamp: 1489046102.20607
   X-Timestamp: 1489046102.20607
   X-Trans-Id: tx73c9382f200d4bd88d866-0058c10a55
   Content-Type: text/plain; charset=utf-8
   ```

   Note: To avoid specifying the option to the swift command, you can use the ~/openrc file from the protocol node. The swift command looks like:

   ```
   source ~/openrc
   swift stat
   ```

2. To create a new account, do the following steps:

   a. Use the **openstack project create** command to create a project.

For example, create the project 'salesproject' in the Default domain using the command:

```
# openstack project create salesproject --domain Default
```

The system displays output similar to the following:

```
+-------------+---------------------------------+
| Field       | Value                           |
+-------------+---------------------------------+
| description | Description is displayed here.  |
| domain_id   | default                         |
| enabled     | True                            |
| id          | ec4a0bff137b4c1fb67c6fe8fbb6a37b|
| is_domain   | False                           |
| name        | salesproject                    |
| parent_id   | default                         |
+-------------+---------------------------------+
```

b. Use the **openstack role add** command to associate roles to the users who need access to the project:

```
 # openstack role add --user admin --project salesproject admin
```

The command does not display any output.

3. To see the new account details, issue the **swift stat** command with the new project value:

```
# swift stat --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3 \
                --os-project-name salesproject \
                --os-project-domain-name Default \
                --os-username admin \
                --os-user-domain-name Default \
                --os-password Passw0rd \
                --auth-version 3
```

The system displays output similar to the following:

```
Account: AUTH_ec4a0bff137b4c1fb67c6fe8fbb6a37b
Containers: 0
Objects: 0
Bytes: 0
X-Put-Timestamp: 1489046460.28283
X-Timestamp: 1489046460.28283
X-Trans-Id: txe9d13765f0c14fe4ad3ce-0058c10bbc
Content-Type: text/plain; charset=utf-8
```

Managing object users, roles, and projects

IBM Spectrum Scale for object storage uses the keystone service for identity management. Keystone provides user authentication and authorization processes.

You can use an external Microsoft Active Directory or LDAP server or a local database as the back-end to store and manage user credentials for user authentication. The authorization details such as relation of users with projects and roles are maintained locally by the keystone server. The customer can select the authentication server to be used. For example, if AD is already configured in the environment and the users who need access to the object store are part of AD, then the customer can configure Keystone with AD as the authentication and authorization back-end.

When the back-end authentication server is AD or LDAP, the user management operations such as creating or deleting a user are the responsibility of the AD/LDAP administrator, who can optionally also be the Keystone server administrator. When local authentication is used for object access, the user management operations are done by the Keystone administrator. In case of authorization, the management tasks such as creating roles, projects, and associating the user with them is done by the Keystone Administrator. The Keystone administration can be done through the Keystone V3 REST API or by using an OpenStack python-based client.

Before you start creating object users, and projects, ensure that Keystone server is configured and the authentication servers are set up properly.

Note:

- If the cluster is reachable from the system, the OpenStack command can be issued from any system.
- If the OpenStack command is run from any of the protocol nodes, then you can use the openrc file to set the required environment that is used by OpenStack commands to manage the Keystone server. The advantage of using the openrc file is that you are not required to enter the following details every time you enter the commands: --os-identity-api-version, --os-username, --os-password, --os-project-domain-name, --os-user-domain-name, --os-domain-id, and --os-auth-url.
- The user create, update, and delete operations are only applicable when local authentication method is used for object access.
- For more information on the Keystone V3 REST API, see the OpenStack API Documentation (developer.openstack.org/api-ref-identity-v3.html).

Creating a new user

When creating a new user in the local database to support local authentication for object access, activate the openrc file located under /root/openrc by default. You can load the openrc profile by running: **'source /root/openrc'** and this will automatically load the required environmental variables into your current location. The results will look similar to this:

```
export OS_AUTH_URL="http://cesobjnode:35357/v3"
export OS_IDENTITY_API_VERSION=symph
export OS_AUTH_VERSION=3
export OS_USERNAME="admin"
export OS_PASSWORD="Passw0rd"
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_PROJECT_DOMAIN_NAME=Default
```

Use the **openstack user create** command and manually enter the parameters as shown in the following example to create new user in the local database to support local authentication for object access.

```
# openstack --os-identity-api-version 3 --os-username admin --os-password
Passw0rd --os-project-domain-name Default --os-user-domain-name Default --osdomain-
id default --os-auth-url http://specscaleswift.example.com:35357/v3 user create --password-prompt
--email newuser1@localdomain.com --domain default newuser1
User Password:
Repeat User Password:
+-----------+--------------------------------------------------------------------------+
| Field     | Value                                                                    |
+-----------+--------------------------------------------------------------------------+
| domain_id | default                                                                  |
| email     | newuser1@localdomain.com                                                 |
| enabled   | True                                                                     |
| id        | 2a3ef80313594572922274bcd70e34d00                                        |
| name      | newuser1                                                                 |
+-----------+--------------------------------------------------------------------------+
```

GUI navigation

To work with this function in the IBM Spectrum Scale GUI, log on to the GUI and select **Object** > **Users**.

Listing users

Use the **openstack user list** command as shown in the following example to list users who are created in the local database:

```
# source $HOME/openrc
# openstack user list
+--------------------------------+----------+
| ID                             | Name     |
```

```
+----------------------------------+----------+
| 2a3ef8031359457292274bcd70e34d00 | newuser1 |
| a95783144edd414aa236a3d1582a3067 | admin    |
+----------------------------------+----------+
```

Changing the password of a user

Use the **openstack user set** command to update the object user details. The following example shows how to change the password:

```
# openstack user set --password Passw0rd newuser2
```

Deleting a user

Use the **openstack user delete** command as shown in the following example to delete the users who are created in the local database:

```
# openstack user delete newuser2
```

Listing user roles

Use the **openstack role list** command as shown in the following example to list the user roles:

```
# openstack role list
+----------------------------------+-------+
| ID                               | Name  |
+----------------------------------+-------+
| ed38022b46094a51918e6e46f87e7290 | admin |
+----------------------------------+-------+
```

Creating a new role

Perform the following steps to create a new user role:

1. Issue the **openstack role create** command to create a new user role:

   ```
   #openstack role create member
   +-----------+----------------------------------------------------------+
   | Field     | Value                                                    |
   +-----------+----------------------------------------------------------+
   | domain_id | None                                                     |
   | id        | 1f14f95826fe4c8590760b3d3e4ce7e0                         |
   | name      | member                                                   |
   +-------+--------------------------------------------------------------+
   ```

2. Verify the newly created role by using the **openstack role list** command:

   ```
   # openstack role list
   +----------------------------------+--------+
   | ID                               | Name   |
   +----------------------------------+--------+
   | 1f14f95826fe4c8590760b3d3e4ce7e0 | member |
   | ed38022b46094a51918e6e46f87e7290 | admin  |
   +----------------------------------+--------+
   ```

GUI navigation

To work with this function in the IBM Spectrum Scale GUI, log on to the GUI and select **Object** > **Roles**.

Assigning a role to a user

Perform the following steps to assign a user role to a user:

1. Issue the **openstack role add** command to assign role to a user as shown in the following example:

   ```
   # openstack role add --user newuser1
   --domain default member
   ```

2. Submit the **openstack role list** command to verify the user role of the user as shown in the following example:

```
# openstack role list --user newuser1
+----------------------------------+--------+
| ID                               | Name   |
+----------------------------------+--------+
| 1f14f95826fe4c8590760b3d3e4ce7e0 | member |
+----------------------------------+--------+
```

Creating a new project, adding a user, and assigning a role to the user

Perform the following steps to create a new project and add a user to the project with a specified role:

1. Submit the **openstack project create** command to create a new project:

```
# openstack project create newproject
+-------------+----------------------------------------------------------------------+
| Field       | Value                                                                |
+-------------+----------------------------------------------------------------------+
| description |                                                                      |
| domain_id   | default                                                              |
| enabled     | True                                                                 |
| id          | 2dfcbdb70b75435fb2015c86d46ffc0b                                     |
| is_domain   | False                                                                |
| name        | newproject                                                           |
| parent_id   | None                                                                 |
+-------------+----------------------------------------------------------------------+
```

2. Submit the **openstack role add** command to add a role to the user as shown in the following example:

```
# openstack role add --user newuser1 --
project newproject member

# openstack role add --user newuser1
--project newproject admin
```

3. Submit the **openstack role list** command to list the user roles as shown in the following example:

```
# openstack role list --user newuser1 --
project newproject
+----------------------------------+--------+------------+----------+
| ID                               | Name   | Project    | User     |
+----------------------------------+--------+------------+----------+
| 1f14f95826fe4c8590760b3d3e4ce7e0 | member | newproject | newuser1 |
| ed38022b46094a51918e6e46f87e7290 | admin  | newproject | newuser1 |
+----------------------------------+--------+------------+----------+
```

Listing endpoints

Use the **openstack endpoint list** command as shown in the following example to view the endpoints that are available:

```
# openstack endpoint list
+------------+-----------+--------------+--------------+---------+-----------+---------------------------------------------------------------------+
| ID         | Region    | Service Name | Service Type | Enabled | Interface | URL                                                                 |
+------------+-----------+--------------+--------------+---------+-----------+---------------------------------------------------------------------+
| c36e..9da5 | RegionOne | keystone     | identity     | True    | public    | http://specscaleswift.example.com:5000                              |
| f4d6..b040 | RegionOne | keystone     | identity     | True    | internal  | http://specscaleswift.example.com:35357                             |
| d390..0bf6 | RegionOne | keystone     | identity     | True    | admin     | http://specscaleswift.example.com:35357                             |
| 2e63..f023 | RegionOne | swift        | object-store | True    | public    | http://specscaleswift.example.com:8080/v1/AUTH_%(tenant_id)s        |
| cd37..9597 | RegionOne | swift        | object-store | True    | internal  | http://specscaleswift.example.com:8080/v1/AUTH_%(tenant_id)s        |
| a349..58ef | RegionOne | swift        | object-store | True    | admin     | http://specscaleswift.example.com:8080                              |
+------------+-----------+--------------+--------------+---------+-----------+---------------------------------------------------------------------+
```

Deleting expired tokens

By default, the Keystone Identity Service stores expired tokens in the database indefinitely. While potentially useful for auditing in production environments, the accumulation of expired tokens considerably increases the database size and might affect the service performance.

Use cron as follows to configure a periodic task on one of the protocol nodes that purges expired tokens hourly or based on the load in your environment.

```
# (crontab -l -u keystone 2>&1 | grep -q token_flush) || \
  echo '@hourly /usr/bin/keystone-manage token_flush >/var/log/keystone/keystone-tokenflush.log 2>&1' \
  >> /var/spool/cron/keystone
```

Deleting the authentication and the ID mapping configuration

Deleting the authentication and ID mapping configuration results in loss of access to data. Before you remove or edit ID mappings, determine how access to data is going to be maintained.

Removing file authentication

Note: You are not allowed to delete both the authentication configuration and the ID mappings at the same time. You need to remove the authentication configuration first and then the ID maps. The system does not allow you to delete the ID maps without deleting the authentication configuration.

1. Issue the **mmuserauth service list** command to see the authentication method that is configured in the system:

   ```
   # mmuserauth service list
   FILE access configuration: LDAP
   PARAMETERS               VALUES
   -------------------------------------------------
   ENABLE_ANONYMOUS_BIND    false
   ENABLE_SERVER_TLS        false
   ENABLE_KERBEROS          false
   USER_NAME                cn=manager,dc=example,dc=com
   SERVERS                  10.0.100.121
   NETBIOS_NAME             eslhnode
   BASE_DN                  dc=example,dc=com
   USER_DN                  ou=people,dc=example,dc=com
   GROUP_DN                 none
   NETGROUP_DN              ou=netgroup,dc=example,dc=com
   USER_OBJECTCLASS         inetOrgPerson
   GROUP_OBJECTCLASS        posixGroup
   USER_NAME_ATTRIB         cn
   USER_ID_ATTRIB           uid
   KERBEROS_SERVER          none
   KERBEROS_REALM           none
   OBJECT access not configured
   PARAMETERS               VALUES
   -------------------------------------------------
   ```

2. Issue the **mmuserauth service remove** command to remove the authentication configuration as shown in the following example:

   ```
   # mmuserauth service remove -—data-access-method file
   mmcesuserauth service remove: Command successfully completed.
   ```

3. Issue the **mmuserauth service list** command to verify whether the authentication configuration is removed:

   ```
   # mmuserauth service list
   FILE access not configured
   PARAMETERS               VALUES
   -------------------------------------------------
   OBJECT access not configured
   PARAMETERS               VALUES
   -------------------------------------------------
   ```

For more information, see *mmuserauth command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Deleting authentication configuration as shown in the previous example does not delete the ID maps. Use the **--idmapdelete** option with the **mmuserauth service remove** command to remove ID maps that are created for user authentication:

```
# mmuserauth service remove --data-access-method file --idmapdelete
mmuserauth service remove: Command successfully completed
```

Removing object authentication

The deletion of ID maps that are used for file access is only applicable when AD with Automatic ID mapping or RFC2307 ID mapping is configured.

Deleting ID maps might also be required in the case of object access. ID map delete option can be used if the system administrator wants to clean up the entire Keystone authentication configuration, including the mapping of users with projects and roles. Cleaning up of ID mapping information results in loss of access to any existing data that is being accessed through the Object Storage interface. Deleting ID mappings deletes user-role-projects mappings as well. Without these mappings, new users are unable to access the old data unless the keystone administrator creates the mapping again for the new user. ID maps are deleted in environments where the object protocol needs to be removed or the entire object store needs to be erased. This is usually done in preproduction or test environments.

If you want to change the authentication method that is already configured for object access, you must remove the authentication method and ID mappings by issuing the **mmuserauth service remove --data-access-method object** and **mmuserauth service remove --data-access-method object --idmapdelete** commands in sequence, as shown in the following example:

```
# mmuserauth service remove --data-access-method object
mmuserauth service remove: Command successfully completed

# mmuserauth service remove --data-access-method object --idmapdelete
mmuserauth service remove: Command successfully completed

# mmuserauth service list
FILE access not configured
PARAMETERS VALUES
-------------------------------------------------
OBJECT access not configured
PARAMETERS VALUES
-------------------------------------------------
```

Note: When you delete the ID maps that are created for file or object access, ensure that all the protocol nodes are in the healthy state. You can view the health status of protocol nodes by using the **mmces state show -a** command.

Listing the authentication configuration

Use the **mmuserauth service list** command to see the authentication method that is configured in the system.

```
# mmuserauth service list
FILE access configuration : LDAP
PARAMETERS VALUES
-------------------------------------------------
ENABLE_SERVER_TLS false
ENABLE_KERBEROS false
USER_NAME cn=manager,dc=example,dc=com
SERVERS 9.122.123.172
NETBIOS_NAME eslhnode
BASE_DN dc=example,dc=com
USER_DN ou=people,dc=example,dc=com
GROUP_DN none
NETGROUP_DN ou=netgroup,dc=example,dc=com
USER_OBJECTCLASS inetOrgPerson
```

```
GROUP_OBJECTCLASS posixGroup
USER_NAME_ATTRIB cn
USER_ID_ATTRIB uid
KERBEROS_SERVER none
KERBEROS_REALM none
OBJECT access not configured
PARAMETERS VALUES
-------------------------------------------------
```

For more information, see the topic *mmuserauth command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Verifying the authentication services configured in the system

Use the **mmuserauth service check** command to check whether the authentication configuration is consistent across the cluster and the required services are enabled and running. This command validates and corrects the authentication configuration files and starts any associated services if needed.

You can check the following authentication details by using the **mmuserauth service check** command:

- --data-access-method {file|object|all} Authentication method.
- [-N|--nodes] {node-list|cesNodes} Authentication configuration on each node. If the specified node is not a protocol node, the check operation gets ignored on that node. If a protocol node is specified, then the system checks configuration on that protocol node. If you do not specify a node, the system checks the configuration of only the current node. To check authentication configuration on all protocol nodes, specify -N cesnodes.
- --server-reachability Verify whether the authentication backend server is reachable. If object is configured with external Keystone server, this check is not performed.
- [-r | --rectify] Rectify the configuration for the specified nodes by copying any missing configuration files or SSL/TLS certificates from another node.

For more information, see the topic *mmuserauth command* in the *IBM Spectrum Scale: Command and Programming Reference* guide.

Example - File authentication check

Issue the **mmuserauth service check** command.
```
# mmuserauth service check --data-access-method file --nodes dgnode3,dgnode2
--server-reachability -r
dgnode2: not CES node. Ignoring...

Userauth file check on node: dgnode3
Checking SSSD_CONF: OK
Checking nsswitch file: OK
Checking Pre-requisite Packages: OK

LDAP servers status
LDAP server 192.168.122.250 : OK
Service 'sssd' status: OK
```

You can use the **id** command to see the list of users and groups fetched from the LDAP server. For example:
```
# id ldapuser2
uid=1001(ldapuser2) gid=1001(ldapuser2) groups=1001(ldapuser2)
```

Example - Object authentication check

Issue the **mmuserauth service check** command.

```
# mmuserauth service check
--server-reachability --data-access-method
object
Userauth object check on node: dgnode3
Checking keystone.conf: OK
LDAP servers status
LDAP server sonash1 : OK
Service 'keystone-all' status: OK
```

Modifying the authentication method

If data already exists or is created with the existing authentication and ID mapping method, it is not recommended to change the authentication or the ID mapping modes. Changing the authentication method also might invalidate the existing ACLs that are applicable to files and directories. ACLs depend on the preexisting users and group IDs.

To modify the authentication method, perform the following steps:

1. List the existing authentication configuration for file and object authentication method by using the **mmuserauth service list** command.

2. Identify the parameters that you need to change. If an authentication method and ID maps are already existing, you must not plan to change the authentication type or ID mapping schemes. When you remove the existing authentication method and ID maps, the user and group of users who were accessing the data cannot access the data anymore.

   The following list provides the parameters that can be modified in each authentication configuration.

   For file authentication:
   - With LDAP authentication, all attributes of the configuration can be modified. When changing authentication servers, ensure that the newly specified servers are the replica of the original servers, otherwise, it might result in loss of access to data.
   - With AD authentication, all attributes of the configuration can be modified. When changing the authentication server, ensure that the newly specified server is a domain controller in the same AD domain that is being served by the original server, otherwise, it might result in loss of access to data. If UNIX ID maps are specified in current configuration and more new AD domains are to be added, it is vital to specify the current list of domains along with the new domains.
   - With NIS authentication, all attributes of the configuration can be modified. When changing servers, ensure that the newly specified servers are serving the same NIS domain as the original servers; otherwise, it might result in loss of access to data.

   For object authentication:

   You can change all options except **--data-access-method** and **--type** parameters.

3. Clean up the existing authentication by using the **mmuserauth service remove** command. Do not specify the **--idmapdelete** option as it results in loss of access to data.

4. Issue the **mmuserauth service create** with the required parameter change; ensuring that you use the same authentication, ID mapping scheme, and associated authentication servers.

5. List the authentication configuration by using the **mmuserauth service list** to verify the change.

6. Ensure that the authentication is consistent across the cluster by using the **mmuserauth service check** command.

Authentication limitations

Consider the following authentication limitations when you configure and manage the IBM Spectrum Scale system:

Object access limitations

The following limitations exist for Active Directory (AD)-based authentication for object access:

- Only single AD server is used. If the configured AD server is down, the Keystone authentication fails.
- Does not support multiple AD Domains.
- Only Windows 2008 R2 and later are supported.
- Authentication is supported only for read access to the AD server. You cannot create a new user and modify or delete an existing user from the IBM Spectrum Scale system. Only the AD server administrator can do these tasks.

The following limitations exist for Lightweight Directory Access Protocol (LDAP)-based authentication for object access:

- Only single LDAP server is used. If the configured LDAP server is down, the Keystone authentication fails.
- Only LDAP servers compatible with LDAP RFC 4511 are supported.
- Authentication is supported only for read access to the LDAP server. You cannot create a new user and modify or delete an existing user from the IBM Spectrum Scale system. Only the LDAP server administrator can do these tasks.

File access limitations

AD based authentication

NFS with server-side group lookup and Active Directory authentication is only supported for Kerberized NFS access. The reason behind this is that obtaining the group membership of a user on a CES node is only possible after authenticating the user authenticated on that node. With SMB, each new session is authenticated initially, which is sufficient to provide that information. With NFS, only Kerberized access can reliably provide the required information when using the Active Directory.

The following limitations exist for AD with automatic ID mapping:

- No support is provided for migrating the internally generated user and group ID maps to an external ID mapping server. If data is stored on the IBM Spectrum Scale system with AD and automatic ID mapping, adding RFC2307 later requires the UIDs and GIDs that are used internally by the IBM Spectrum Scale system match the UIDs and GIDs stored in RFC2307. Matching is not possible if conflicting UIDs and GIDs are already stored in RFC2307. To avoid potential conflicts, configure the IBM Spectrum Scale system by using AD and RFC2307 from the beginning.
- Although AD along with automatic ID mapping can be used to have the same ID maps between systems that are in AFM relationship, this configuration is not a complete replacement for RFC2307. This configuration can be used in a predominantly SMB only setup, where NFS users are not already present in the environment. If NFS users are preexisting in the customer environment and these users intend to access the data with SMB users, then RFC2307 is mandatory.
- When AD-based authentication is used, SMB protocol access is kerberized by default. Access the system by using the netbios name that is specified in the command.

The following limitations exist for AD with RFC2307:

- Enabling RFC2307 for a trusted domain requires a two-way trust between the native and the trusted domains.
- To access the IBM Spectrum Scale system, users and groups must have a valid UID/GID assigned to them in AD. For user access, the windows group membership is evaluated on the IBM Spectrum Scale system. Hence, accessing a user's primary group is considered as the Microsoft Windows Primary group and not the UNIX primary group that is listed in the UNIX attribute tab in the user's properties. Therefore, the user's primary Microsoft Windows group must be assigned with a valid GID.

- The `mmuserauth service create` command does not check the two-way trust between the native domain and the RFC2307 domain that is required for ID mapping services to function properly. The customer is responsible for configuring the two-way trust relationship between these domains. The customer is responsible for assigning UIDs to users and GIDs to groups. The command does not return an error if a UID or GID is not assigned.

LDAP-based authentication

The following limitations exist for LDAP-based authentication:
- Users with the same user name from different organizational units under the specified baseDN in the LDAP server are denied access to SMB shares irrespective of the LDAP user suffix and LDAP group suffix values configured on the system.
- If multiple LDAP servers are specified during configuration, at any point in time, only one LDAP server is used.
- LDAP referrals are not supported.
- ACL management through windows clients is not supported.
- Only LDAP servers that implement RFC2307 schema are supported.

General limitations for file access

The following general limitations exist:
- When the SMB service is stopped on a protocol node, with any AD-based authentication method, the NFS-based access is also affected on that protocol node.
- When Microsoft Active Directory (AD) is used as an authentication system, the IBM Spectrum Scale system supports only the NetBIOS logon name for authentication and not the User Principle Name (UPN). Active Directory replaces some of the special characters that are used in the UPN with the underscore character (hexadecimal value 0x5F) for the related NetBIOS logon name of the user. For the complete list of the special characters that are replaced in the NetBIOS logon name, see Microsoft Active Directory documentation. Follow these steps to locate the NetBIOS logon name for an Active Directory domain user:
  1. From the Windows Start menu, select Administrative Tools > Active Directory Users and Computers.
  2. Right-click the Active Directory Domain user for which you require the NetBIOS logon name.
  3. Select Properties > Account Tab and check the value of the User logon name field (pre-Windows 2000).
- Authentication configuration commands restart the IBM Spectrum Scale protocol services such as SMB and NFS. The protocol services resume a few seconds after an authentication configuration command completes.
- For file data access, switching or migrating from one authentication method to another is not supported, because it might lead to loss of access to the data on the system.
- The IBM Spectrum Scale system does not support authentication servers (AD, LDAP, and NIS) that are running on virtual machines that are stored on an SMB or NFS export. The IBM Spectrum Scale system requires the authentication server to be running while you are configuring authentication and while the server is handling connection requests over protocols. The virtualizer cannot boot the authentication server unless the protocols are configured for authentication and data is ready to be served over the exports.
- The length of a user name or a group name of the users and group of users who need to access the data cannot be more than 32 characters.
- The NFSV4 clients must be configured with the same authentication and ID mapping server as the IBM Spectrum Scale system. The IBM Spectrum Scale system does not support an NFSV4 client that is configured with different authentication and ID mapping servers.

- AIX clients follow a different methodology to integrate with AD, and hence, NFSV4-based access from AIX clients to IBM Spectrum Scale is not supported when CES services are configured for AD and variations of AD-based authentication schemes.
- Based on the hardware platform that the protocol nodes are configured on, consider the group ID resolution in relation to the limitation that is described in the IBM Spectrum Scale FAQ. For more information, see IBM Spectrum Scale FAQs.
- With regard to AD-based authentication scheme, the following considerations apply to configuring an NFS server to look up group membership information for an accessing NFS user:
  - The server-side group lookup functionality, which is enabled by setting the `MANAGE_GIDS` flag in the NFS configuration, works only after the user makes a valid authentication connection over CIFS.
  - You must make a valid authentication connection to the protocol node that serves the public IP from which the NFS export is to be mounted.
  - If the group membership of the user on an AD server changes, you must make a new valid CIFS connection to the protocol node that serves the public IP from which the NFS export is to be mounted. This new connection reflects the changes on the protocol node of the CES cluster.
  - It is a good practice to make a valid authentication connection over CIFS to all the protocol nodes that participate in group membership evaluations. This practice results in uniform membership evaluations on all the protocol nodes of the CES cluster.
- To use NFSV4 ID mapping, you must set the NFS ID map domain on the IBM Spectrum Scale protocol nodes and you must configure the same NFS ID map domain on every NFS client. The following example demonstrates how to configure NFSV4 ID mapping.
  1. Issue the **mmnfs config list** command.

     The system displays the following output, which shows that the ID map domain is not set:

     ```
     Idmapd Configuration
     ====================
     ====================
     ```
  2. Enter the following command to set the NFS ID map domain:

     ```
     mmnfs config change IDMAPD_DOMAIN=MY_IDMAP_DOMAIN
     ```
  3. Issue the **mmnfs config list** command to verify that the ID map domain is set.

     The system displays this output:

     ```
     Idmapd Configuration
     =======================
     DOMAIN: MY_IDMAP_DOMAIN
     =======================
     ```

Chapter 18. Managing protocol data exports

You can manage the data exports that you have created using NFS, SMB, and Object.

Managing SMB shares

All SMB administration commands can be run from any cluster node including non-CES nodes. However, the latency of the administration command execution on a CES node is lower because administrative changes are made immediately. Use the following information to manage SMB shares in IBM Spectrum Scale.

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Protocols** > **SMB Shares**.

Creating SMB share

Use the following information to create an SMB share:

1. Create the directory to be exported through SMB:

    Note: IBM recommends an independent fileset for SMB shares.

    Create a new independent fileset with these commands:

    ```
    mmcrfileset fs01 fileset --inode-space=new
    mmlinkfileset fs01 fileset -J /gpfs/fs01/fileset
    ```

    If the directory to be exported does not exist, create the directory first by running the following command:

    ```
    mkdir /gpfs/fs01/fileset/smb
    ```

2. The recommended approach for managing access to the SMB share is to manage the ACLs from a Windows client machine. To change the ACLs from a Windows client, change the owner of the share folder to a user ID that will be used to make the ACL changes by running the following command:

    ```
    chown 'DOMAIN\smbadmin' /gpfs/fs01/fileset/smb
    ```

3. Create the actual SMB share on the existing directory:

    ```
    mmsmb export add smbexport /gpfs/fs01/fileset/smb
    ```

    Additional options can be set during share creation. For a list of all the supported SMB options, see *mmsmb command* in the *IBM Spectrum Scale: Command and Programming Reference*.

4. Verify that the share has been created:

    ```
    mmsmb export list
    ```

5. Access the share from a Windows client using the user ID that has been previously made the owner of the folder.

6. Right-click the folder in the Windows Explorer, open the **Security** tab, click **Advanced**, and modify the Access Control List as required.

    Note: An SMB share can only be created when the ACL setting of the underlying file system is **-k nfsv4**. In all other cases, **mmsmb export add** will fail with an error.

    See "Authorizing protocol users" on page 316 for details and limitations.

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Protocols** > **SMB Shares**.

Changing SMB share configuration

Use the following information to change the SMB share configurations.

For the documentation of all supported options, see *mmsmb command* in the *IBM Spectrum Scale: Command and Programming Reference.*

To see a list of supported configuration options for SMB shares, run the command:
```
mmsmb export change --key-info supported
```

For example, to change the descriptive comment for a share, run the command:
```
mmsmb export change smbshare --option 'comment=Project X export'
```

To list the configuration of all SMB shares, run the command:
```
mmsmb export list --all
```

Note: Changes to SMB share configurations only apply to client connections that have been established after the change has been made.

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Protocols** > **SMB Shares**.

Creating SMB share ACLs

The SMB protocol supports a separate level of ACLs that can be optionally added to an SMB share.

For more information, see Managing ACLs of SMB exports using MMC.

For details, see the information about managing the SMB share ACLs from a Windows client through the MMC.

Removing SMB shares

To remove an SMB share, use the **mmsmb** command. Use the following information to remove SMB shares.

To remove an SMB share:
1. Run the following command:
   ```
   mmsmb export remove smbexport
   ```
2. Verify that the export has been removed by listing the configured SMB share again:
   ```
   mmsmb export list
   ```

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Protocols** > **SMB Shares**.

Listing SMB shares

To list the SMB shares, run the following command:
```
mmsmb export list
```

Managing SMB shares using MMC

Microsoft Management Console (MMC) is a Windows tool that can be used to do basic configuration tasks on an SMB server. These tasks include administrative tasks such as listing or closing the connected users and open files, and creating and manipulating SMB shares. You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for managing SMB shares on the IBM Spectrum Scale cluster.

Attention: Listing a large number of entities (thousands of files, connections, locks, etc.) using Microsoft Management Console (MMC) might take a very long time and it might impact the performance of the file server. In these cases, it is recommended to use server-side administration tools.

Ensure that the following tasks are complete before you manage SMB shares:
- IBM Spectrum Scale is installed and configured.
- The SMB protocol is enabled and healthy SMB services are running on all protocol nodes.
- Required SMB shares are created and mounted from the Windows client.
- Microsoft Active Directory (AD) based authentication is set up. This includes:
  - Cluster nodes and client are domain members.
  - The client on which Microsoft Management Console (MMC) is running is a domain member.
  - Accurate DNS information is configured. If active sessions are listed, MMC tries to do a reverse pointer record lookup with DNS for every session (client IP), and if that fails then MMC hangs.
  - Involved NetBIOS names can be resolved using DNS.

For using the Shared Folders Microsoft Management Console (MMC) snap-in, you must be a member of the local administrators group of the cluster. After joining the cluster to an AD domain, only the domain admins group is a member of the administrators group of the cluster.

To add other users who can use the Shared Folders Microsoft Management Console (MMC) snap-in:
1. Connect to MMC as a user that is a member of the domain admins group.
2. Navigate to **System Tools** > **Local Users and Groups** and add a user to the local administrators group.

For more information, see the Microsoft Management Console documentation.

The following MMC features are not supported for managing SMB shares on the IBM Spectrum Scale cluster:
- Audit of MMC read operations
- Event viewer
- Setting max connections per share

Connecting to SMB shares by using MMC

You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for connecting to SMB shares on the IBM Spectrum Scale cluster.
1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.
2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.
3. In the left pane, click **Shares**. All SMB shares are listed in the right pane.

Note: If there is a permissions related error when you click **Shares**, verify that you are a member of the local administrators group of the cluster. For more information, see "Managing SMB shares using MMC" on page 225.

Creating SMB shares using MMC

You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for creating SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.
2. Connect to the server on which you want to create SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.
3. In the left pane, right-click **Shares** and then click **New Share**. The Create A Shared Folder wizard opens.

   Note: If there is a permissions related error when you click **Shares**, verify that you are a member of the local administrators group of the cluster. For more information, see "Managing SMB shares using MMC" on page 225.
4. In the Create A Shared Folder wizard, click **Next**.
5. In the **Folder path** field, enter the share path and click **Next**.

   Note: The directory for the SMB has to already exist in the file system.
6. Enter the SMB share name and description, select the required offline setting, and then click **Next**.
7. Select the required SMB share permission setting and click **Finish**.

Modifying or removing SMB shares using MMC

You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for modifying or removing SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.
2. Connect to the server on which you want to create SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.
3. In the left pane, click **Shares**. All SMB shares are listed in the right pane.

   Note: If there is a permissions related error when you click **Shares**, verify that you are a member of the local administrators group of the cluster. For more information, see "Managing SMB shares using MMC" on page 225.
4. Do one of the following steps depending on whether you want to modify or remove SMB shares:
   - To modify an SMB share:
     a. In the right pane, right-click the SMB share that you want to modify, and then click **Properties**.
     b. Modify the properties as required and click **OK**.
   - To remove an SMB share:

a. In the right pane, right-click the SMB share that you want to remove, and then click **Stop Sharing**.

Managing ACLs of SMB shares using MMC

You can use Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for managing access control lists (ACLs) of SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.

2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.

3. In the left pane, click **Shares**. All SMB shares are listed in the right pane.

   Note: If there is a permissions related error when you click **Shares**, verify that you are a member of the local administrators group of the cluster. For more information, see "Managing SMB shares using MMC" on page 225.

4. In the right pane, right-click the SMB share for which you want to view or change the permissions and then click **Properties**.

5. You can do one of the following:
   - To view the permissions a user or a group has for the SMB share, on the **Share Permissions** tab, under the "Group or user names" pane, click on the user name or the group name.
     The permissions are displayed in the "Permissions for" pane.
   - To change the permissions a user or a group has for the SMB share, on the **Security** tab, under the "Group or user names" pane, click on the user name or the group name and then click **Edit**.

     Note: Changes affect only the SMB share, not the ACL in the file system of the exported directory.

     For information on permissions that you can change, see documentation for the Shared Folders Microsoft Management Console (MMC) snap-in.

Modifying offline settings of SMB shares using MMC

You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for modifying offline settings of SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.

2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.

3. In the left pane, click **Shares**. All SMB shares are listed in the right pane.

   Note: If there is a permissions related error when you click **Shares**, verify that you are a member of the local administrators group of the cluster. For more information, see "Managing SMB shares using MMC" on page 225.

4. In the right pane, right-click the SMB share whose offline settings you want to modify, and then click **Properties**.

5. On the **General** tab, click **Offline Settings**.
6. In the Offline Settings window, configure the offline settings of the SMB share. For information on offline settings that you can configure, see documentation for the Shared Folders Microsoft Management Console (MMC) snap-in.

Viewing active connections to SMB shares using MMC

You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for viewing active connections to SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.
2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.
3. In the left pane, click **Sessions**. All active connections to SMB shares are listed in the right pane.

Disconnecting active connections to SMB shares using MMC

You can use Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for disconnecting active connections to SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.
2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.
3. In the left pane, click **Sessions**. All active connections to SMB shares are listed in the right pane.
4. In the right pane, right-click the connection that you want to close and then click **Close Session**.
   Attention: If connections are forced to close, data loss might occur for open files on the connections getting closed.
5. Click **OK** to confirm.

Viewing open files in SMB shares using MMC

You can use Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for viewing open files in SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:
   a. Click **Start** > **Run**.
   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.
2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:
   a. Click **Action** > **Connect to another computer**.
   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.
3. In the left pane, click **Open Files**. All open files in SMB shares are listed in the right pane.

Viewing the number of locks on files in SMB shares using MMC

You can use the Shared Folders Microsoft Management Console (MMC) snap-in on Microsoft Windows clients for viewing the number of locks on open files in SMB shares on the IBM Spectrum Scale cluster.

1. Open the Shared Folders Microsoft Management Console (MMC) snap-in:

   a. Click **Start** > **Run**.

   b. Type `fsmgmt.msc` and click **OK**.

   The Shared Folders Microsoft Management Console (MMC) snap-in opens.

2. Connect to the IBM Spectrum Scale cluster that has the SMB shares:

   a. Click **Action** > **Connect to another computer**.

   b. Type the IP address of the server you want to connect to in the **Another computer** field and click **OK**.

3. In the left pane, click **Open Files**. All open files in SMB shares are listed in the right pane.

4. In the right pane, view locks on a file under the **# Locks** column.

   The number of locks is displayed under the **# Locks** column and the type of locks is displayed under the **Open Mode** column.

Managing NFS exports

Use the following information to manage NFS exports in IBM Spectrum Scale.

Creating NFS exports

To add an NFS export, use the **mmnfs** export add command.

1. If the directory to be exported does not exist, create the directory by running the following commands:

   ```
   mmcrfileset fs01 fileset --inode-space=new
   ```

   ```
   mmlinkfileset fs01 fileset -J /gpfs/fs01/fileset
   ```

   For more details, see *mmcrfileset command* and *mmlinkfileset command* in *IBM Spectrum Scale: Command and Programming Reference*.

   Note: We recommend an independent fileset for NFS exports.

2. Adjust the ownership and permissions of the folder as required.

   Use the GPFS ACL's with **mmgetacl** and **mmputacl** to set the correct ownership and the access permission.

   Additional options can be set during export creation. For the documentation of all supported options, see *mmnfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

3. Create the NFS export using the following command:

   ```
   mmnfs export add /gpfs/fs01/fileset
   ```

4. To export the fileset fset_1 to specific set of hosts that fall in the 255.255.0.0 subnet, issue this command:

   ```
   mmnfs export add /gpfs/fs01/fset_1 --client 10.1.0.0/16
   ```

5. To export the fileset fset_2 to specific set of hosts that fall in the 255.255.255.0 subnet, issue this command:

   ```
   mmnfs export add /gpfs/fs01/fset_2 --client 10.1.1.0/24
   ```

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Protocols** > **NFS Exports**.

Changing NFS export configuration

After an NFS export is created, the export attributes can be changed by using the `mmnfs export change` command.

For the documentation of all supported options, see *mmnfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

For example, to grant another client IP address access to the NFS export, run the following command:
```
mmnfs export change /gpfs/fs01/nfs --nfsadd 10.23.23.23
```

After the change is made, verify the configuration by running the following command:
```
mmnfs export list
```

The system displays output similar to this:
```
Path Delegations Clients
------------------------------
/gpfs/fs01/nfs none 10.23.23.21
/gpfs/fs01/nfs none 10.23.23.22
/gpfs/fs01/nfs none 10.23.23.23
```

Removing NFS exports

To remove an NFS export, use the `mmnfs export remove` command.

To remove an NFS export, follow these steps:
1. Specify the following command:
   ```
   mmnfs export change /gpfs/fs01/fset1 --nfsremove "192.168.0.28"
   ```

   The system displays output similar to the following:
   ```
   The NFS export was deleted successfully.
   ```
2. Verify that the export is removed by listing the configured NFS exports. Specify:
   ```
    mmnfs export list
   ```
3. If you are exporting multiple clients, specify:
   ```
   mmnfs export list
   --------------------
   /gpfs/fs01/fset1 NONE 192.0.2.0
   /gpfs/fs01/fset1 NONE 192.0.2.1
   /gpfs/fs01/fset1 NONE 192.0.2.2
   ```

   Now in order to remove the 192.0.2.0 export value, specify:
   ```
   mmnfs export change --nfsremove "192.0.2.0"
   ```

   Note: CES NFS does not restart NFS services if the export is removed dynamically. Refer to the `mmnfs` command for more information

Listing NFS exports

To list the NFS exports, enter the following command:
```
mmnfs export list
```

The system displays output similar to the following:
```
Path            Delegations Clients
----------------------------------------
/gpfs/FS1/fset_1 NONE       *
/gpfs/FS1/fset_2 NONE       10.1.0.0/16
/gpfs/FS1/fset_2 NONE       @host_n87_n88
```

GUI navigation for NFS exports

Use the following information to manage NFS exports in IBM Spectrum Scale.

To work with the NFS exports function in the GUI, log on to the IBM Spectrum Scale GUI and select **Protocols** > **NFS Exports**.

Making bulk changes to NFS exports

The `mmnfs export load` command can be used to make bulk changes to existing NFS Exports as an alternative to using the `mmnfs export change` command.

Since the existing command to change an NFS export, `mmnfs export change`, will require a few seconds runtime for every invocation of the command, an alternate method is provided to facilitate bulk changes to the NFS configuration.This procedure can be used, for example, to quickly and easily add additional NFS clients to an export, or to change NFS export attributes, on a per client basis, for any existing NFS client definition.

Note: The `mmnfs export load <NFS_exports_config_file>` command will overwrite the existing NFS exports configuration on every CES node.

Existing NFS export

If there is at least one existing NFS export, use the following procedure to make changes to an NFS exports configuration file:

1. Check that there is at least one existing NFS export by issue the following command:

   ```
   mmnfs export list -Y | grep nfsexports | grep -v HEADER
   ```
2. Check that there is at least one CES node in the cluster:

   ```
   mmces node list -Y | grep -v HEADER
   ```
3. Check that NFS is enabled:

   ```
   mmces service list -Y | grep NFS:enabled
   ```
4. Log into a CES node:

   ```
   ssh `mmces node list -Y | grep -v HEADER | tail -1 | awk -F':' '{print$8}'
   ```
5. Start NFS (if not already started) on the CES node:

   ```
   mmces service start NFS
   ```
6. Copy the existing NFS exports configuration file to /tmp:

   ```
   cp -pr /var/mmfs/ces/nfs-config/gpfs.ganesha.exports.conf /tmp/gpfs.ganesha.exports.conf
   ```
7. Make a backup copy of the original NFS exports configuration file:

   ```
   cp -pr /tmp/gpfs.ganesha.exports.conf /tmp/gpfs.ganesha.exports.conf.bak
   ```
8. Manually edit /tmp/gpfs.ganesha.exports.conf:

   ```
   vim /tmp/gpfs.ganesha.exports.conf
   ```
9. When making changes, observe the following guidelines for attributes and values (subject to change at the discretion of the IBM Spectrum Scale software development team):

   ```
   # EXPORT Only Options (One EXPORT block per Path):
   EXPORT {
   Path="<value>"; # must be unique
   Pseudo="<value>"; # must be unique; usually same as Path
   Tag="<value>"; # must be unique; usually same as Path
   Export_id=<value>; # must be unique
   MaxRead=<value>;
   MaxWrite=<value>;
   PrefRead=<value>;
   PrefWrite=<value>;
   PrefReaddir=<value>;
   ```

```
MaxOffsetWrite=<value>;
MaxOffsetRead=<value>;
Filesystem_id=<value>;
UseCookieVerifier=<value>;
Attr_Expiration_Time=<value>;
Delegations=none;

...
} # encloses EXPORT block containing FSAL and one or more CLIENT blocks
# EXPORT Option Values:
["PATH"]="MANDATORY=yes;TYPE=string;DEFAULT=no_default"
["PSEUDO"]="MANDATORY=yes;TYPE=string;DEFAULT=no_default"
["TAG"]="MANDATORY=yes;TYPE=string;DEFAULT=no_default"
["EXPORT_ID"]="MANDATORY=yes;TYPE=value;MIN=1;MAX=65535;DEFAULT=no_default"
["MAXREAD"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=67108864;DEFAULT=1048576"
["MAXWRITE"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=67108864;DEFAULT=1048576"
["PREFREAD"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=67108864;DEFAULT=1048576"
["PREFWRITE"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=67108864;DEFAULT=1048576"
["PREFREADDIR"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=67108864;DEFAULT=1048576"
["MAXOFFSETREAD"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=18446744073709551615;DEF
AULT=18446744073709551615"
["MAXOFFSETWRITE"]="MANDATORY=yes;TYPE=value;MIN=512;MAX=18446744073709551615;DE
FAULT=18446744073709551615"
["FILESYSTEM_ID"]="MANDATORY=yes;TYPE=value;MIN=0;MAX=18446744073709551615;DEFAU
LT=666.666"
["USECOOKIEVERIFIER"]="MANDATORY=yes;TYPE=bool;DEFAULT=false"
["ATTR_EXPIRATION_TIME"]="MANDATORY=yes;TYPE=value;MIN=0;MAX=360;DEFAULT=60"
# FSAL Only Options (One FSAL block per EXPORT block):
FSAL {
Name=GPFS;
}
# CLIENT Only Options (One or more CLIENT blocks per EXPORT block):
CLIENT {
Clients=<value>;
Access_Type=<value>;
Protocols=<value>;
Transports=<value>;
Anonymous_uid=<value>;
Anonymous_gid=<value>;
SecType=<value>;
PrivilegedPort=<value>;
Manage_Gids=<value>;
Squash=<value>;
NFS_Commit=<value>;
Delegations=none;
}
# CLIENT Option Values:
["CLIENTS"]="MANDATORY=yes;TYPE=string;DEFAULT=*"
["ACCESS_TYPE"]="MANDATORY=yes;TYPE=enum;LIST=none,RW,RO,MDONLY,MDONLY_RO;DEFAUL
T=RO"
["PROTOCOLS"]="MANDATORY=yes;TYPE=enum;LIST=3,4,NFS3,NFS4,V3,V4,NFSv3,NFSv4;DEFA
ULT=3,4"
["TRANSPORTS"]="MANDATORY=yes;TYPE=enum;LIST=UDP,TCP;DEFAULT=TCP"
["ANONYMOUS_UID"]="MANDATORY=yes;TYPE=value;MIN=-
2147483648;MAX=4294967295;DEFAULT=-2"
["ANONYMOUS_GID"]="MANDATORY=yes;TYPE=value;MIN=-
2147483648;MAX=4294967295;DEFAULT=-2"
["SECTYPE"]="MANDATORY=yes;TYPE=enum;LIST=none,sys,krb5,krb5i,krb5p;DEFAULT=sys"
["PRIVILEGEDPORT"]="MANDATORY=yes;TYPE=bool;DEFAULT=false"
["MANAGE_GIDS"]="MANDATORY=yes;TYPE=bool;DEFAULT=false"
```

```
["SQUASH"]="MANDATORY=yes;TYPE=enum;LIST=root,root_squash,rootsquash,all,all_squ
ash,allsquash,no_root_squash,none,noidsquash;DEFAULT=root_squash"
["NFS_COMMIT"]="MANDATORY=yes;TYPE=bool;DEFAULT=false"
```

10. Load the changes to the NFS exports config file (this will restart NFS on every CES node on which NFS is currently running):

```
mmnfs export load /tmp/gpfs.ganesha.exports.conf
```

> **Note:** The **mmnfs export load** command will conduct a check of the exports configuration file. If the following message is displayed, check the syntax of the NFS exports configuration file, focusing on the changes made in the previous step and try again:

```
mmnfs export load. The syntax of the NFS export configuration file to load is not correct:
 /tmp/gpfs.ganesha.exports.conf.
```

11. Verify changes to the NFS configuration via the **mmnfs export list** command:

```
mmnfs export list -Y
```

If a long listing of all NFS exports is desired, use a keyword with the **-n** option. For example, with /gpfs as the keyword (/gpfs is the root of each NFS file system in this case):

```
[11:00:48] xxxxx:~:% mmnfs export list -Y -n /gpfs

mmcesnfslsexport:nfsexports:HEADER:version:reserved:reserved:Path:Delegations:Cl
ients:Access_Type:Protocols:Transports:Squash:Anonymous_uid:Anonymous_gid:SecTyp
e:PrivilegedPort:DefaultDelegations:Manage_Gids:NFS_Commit:

mmcesnfslsexport:nfsexports:0:1:::/gpfs/fs1/fset1:none:10.0.0.1:RO:3,4:TCP:NO_RO
OT_SQUASH:-2:-2:SYS:FALSE:none:FALSE:FALSE:

mmcesnfslsexport:nfsexports:0:1:::/gpfs/fs1/fset1:none:*:RW:3,4:TCP:ROOT_SQUASH:
-2:-2:SYS:FALSE:none:FALSE:FALSE:
```

No existing NFS export

1. Check that there is not an existing NFS export by issuing the following command:

```
mmnfs export list -Y | grep nfsexports | grep -v HEADER
```

2. Create NFS exports (adding the first export will restart NFS on every CES node on which NFS is currently running; adding additional exports will not restart NFS):

```
mmnfs export add <export>
```

Multiprotocol exports

Exports for SMB and NFS protocols can be configured so that they have access to the same data in the GPFS file system.

To export data via NFS and SMB , first create an export for one protocol using the appropriate GPFS command (for example, **mmnfs export add**). In order to export the same GPFS path via a second protocol, simply create another export using the protocol-specific export management command (for example, **mmsmb export add**).

The operations of adding and removing exports do not delete any data in the GPFS file system, and removal of exports does not change the data in the GPFS file system. If at a later time access to a GPFS file system for a specific protocol needs to be removed, this can be done via the corresponding command. It also does not impact access to the same data configured via another protocol.

For more information, see the *Unified file and object access overview* topic in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Multiprotocol export considerations

Exports for SMB and NFS protocols can be configured so that they have access to the same data in the file system. In addition, the data can be accessed directly in the file system on the cluster nodes. When configuring access to the same GPFS file system via both the NFS and SMB protocols, certain limitations apply.

These restrictions apply to the general areas of file locking (including share reservation and lock semantics), recovery (reclaim), and cross-protocol notifications.

Access Control Lists (ACLs): In IBM Spectrum Scale, there is a single common ACL per file or directory in the cluster file system that is used for POSIX, NFS, and SMB access. The SMB server converts each Windows ACL into an NFS4 ACLs for the corresponding file system object.

Shared access (share modes, share reservations): Share modes are feature of the SMB protocol that allows clients to announce what type of parallel access should be allowed by other clients while the file is open. NFSv4 share reservations are the equivalent of SMB share modes for the NFS protocol. There is no equivalent in NFSv3 but IBM Spectrum Scale allows the SMB server to propagate the share modes into the cluster file system so that NFS clients can honor share modes on commonly used files. The corresponding SMB option is `gpfs:sharemodes`.NFSv4 share reservations are currently not supported.

Note that disabling the SMB option `gpfs:sharemodes` can result in data integrity issues, as SMB application can rely on the enforcement of exclusive access to data to protect the integrity of a file's data. As the SMB file server also does sharemode checks internally, `gpfs:sharemodes` can safely be disabled for data that is only accessed through the SMB protocol.

The important point for POSIX and NFS applications is that file system sharemodes can result in the open() or unlink() system calls to return EACCES. Applications must be prepared to handle this situation.

Details for the interaction of SMB with the file system sharemodes:

When an SMB client requests to open a file, it must specify the allowed share modes. The share modes are specified with the FILE_SHARE_READ, FILE_SHARE_WRITE and FILE_SHARE_DELETE flags; each one of those indicate which parallel access is allowed while the SMB client has the file open.
- If FILE_SHARE_READ is not allowed and another application requests to access the same file for reading data (through the POSIX system call open() with the O_RDONLY flag) then the system call fails with the EACCES error code.
- If FILE_SHARE_WRITE is not allowed and another application requests to access the same file for reading data (through the POSIX system call open() with the O_WRONLY flag), then the system call fails with the EACCES error code.
- If FILE_SHARE_DELETE is not allowed and another application requests to delete the file through the unlink() system call, the system call fails with the EACCES error code.

If another application already has the file open for reading and writing, and the specified share mode from the SMB client conflicts with the existing open, then the SMB client cannot open the file and a "sharing violation" error is returned back to the SMB client.

Share modes in the file system are only enforced if the file is actually opened for READ, EXECUTE, WRITE or APPEND access from the SMB client.

Another limitation of the share mode enforcement in the file system is that it is not possible to grant parallel FILE_SHARE_READ and FILE_SHARE_WRITE access, while not granting FILE_SHARE_DELETE access. In this case, the file system does not enforce that the FILE_SHARE_DELETE restriction and the file can still be deleted.

These limitations only apply to enforcement of sharemodes in the file system. The SMB server also performs internal sharemode checks and handles the sharemode correctly for all SMB access.

Note: The CES NFS server keeps the files accessed by NFSv3 open for a while for performance reasons. This might lead to conflicts during concurrent SMB access to these files. You can use the following command to find out whether the NFS server holds the specified file open:

```
ls /proc/$(pidof ganesha.nfsd)/fd -l | grep <file-name>
```

Opportunistic Locking: Oplocks are a feature of the SMB protocol that allows clients to cache files locally on the client. If the SMB server is set to propagate oplocks into the cluster file system (gpfs:leases), other clients (NFS, POSIX) can break SMB oplocks. NFS4 delegations are currently not supported.

Byte-range locks: Byte-range locks from SMB clients are propagated into the cluster file system if the SMB option "posix locking" is true. In that case, POSIX and NFS clients are made aware of those locks. Note that for Windows byte-range locks are mandatory whereas for POSIX they are advisory.

File change notifications: SMB clients can request notifications when objects change in the file system. The SMB server notifies its clients about the changes. The notifications include changes that are triggered by POSIX and NFS clients in the directory for which notifications are requested, but not in its subdirectories, if they are done on any CES node. File changes initiated on non-CES cluster nodes will not trigger a notification.

Grace period: The grace period allows NFS clients to reclaim their locks and state for a certain amount of time after a server failure. SMB clients are not aware of NFS grace periods. If you expect a lot of contention between SMB and NFS, NFSv4 reclaims might fail.

Multiprotocol access of protocol exports is only allowed between NFSV4 and SMB. That is, you cannot access the same export by using both NFSV3 and SMB protocols. The reason is that SMB clients typically request exclusive access to a file which does not work with the CES NFS server that keeps files accessed through NFSv3 open.

Chapter 19. Managing object storage

Use the following information to use and manage the IBM Spectrum Scale for object storage features.

Understanding and managing Object services

Use the following information to manage services related to IBM Spectrum Scale for object storage.

IBM Spectrum Scale uses the `mmces service` command to enable, start, stop, or disable Object services on all protocol nodes.

The enable and disable operations are cluster-wide operations. To enable or disable the Object protocol, use `mmces service [enable | disable] OBJ`. The Object protocol must have been initially configured using the `mmobj swift base` command before it can be enabled in the cluster.

CAUTION:
Disabling the object service unconfigures the Object protocol and discards OpenStack Swift configuration and ring files from the CES cluster. If Openstack Keystone configuration is configured locally, disabling object storage also discards the Keystone configuration and database files from the CES cluster. However, to avoid accidental data loss, the associated filesets used for the object data are not automatically removed during disable. The filesets for the object data and any filesets created for optional object storage policies need to be removed manually. If you plan to re-enable the object protocol after disabling it, it is recommended that you access the repository during object disablement in order to reset to the original default object configuration. For enabling the object service subsequently, either different fileset names need to be specified or the existing filesets need to be cleaned up. For information on cleaning up the object filesets, see the steps *"Remove the fileset created for object"* **and** *"Remove any fileset created for an object storage policy"* **(if applicable) in the** *Cleanup procedures required if reinstalling with the spectrumscale installation toolkit* **topic of** *IBM Spectrum Scale: Concepts, Planning, and Installation Guide***.**

Note: To disable the object protocol, first remove the object authentication. For complete usage information, see the *mmuserauth command* in the *IBM Spectrum Scale: Command and Programming Reference*.

In addition, enabled Object service can be started and stopped on individual nodes or cluster-wide.

To start or stop the Object protocol cluster-wide. use -a flag `mmces service [start | stop] OBJ -a`.

To start or stop the Object protocol on individual nodes, use `mmces service [start | stop] OBJ -N <node>`.

Attention: If object services on a protocol node are stopped by the administrator manually, access to object data might be impacted unless the CES IP addresses are first moved to another node. There are multiple ways to accomplish this, but the simplest is to suspend the node. After suspending a node, CES automatically moves the CES IPs to the remaining nodes in the cluster. However, doing this suspends operation for all protocols running on that protocol node.

If you want to stop object services on a protocol node, you can use the following steps:
1. Suspend CES operations on the protocol node using the `mmces node suspend` command.
2. View the CES IP addresses on that node using the `mmces address list` command and verify that all CES IP addresses have been moved to other protocol nodes.
3. Stop the object services using the `mmces service stop OBJ` command.

Note: All object services must be controlled by using only the mmces service start/stop and systemctl commands. The use of swift-init command is not supported and might cause your system to operate incorrectly.
Performing these steps ensures that object functionality is available on other nodes in the cluster.

To restore object services on that protocol node, you can use the following steps:

1. Resume CES operations on the protocol node using the `mmces node resume` command.
2. View the CES IP addresses on that node using the `mmces address list` command and verify that all CES IP addresses have been moved to that protocol node.
3. Start the object services using the `mmces service start OBJ` command.

Use the **mmces service list** command to list the protocols enabled on IBM Spectrum Scale. List a verbose output of object services running on the local node using the **-v** flag as shown in the following example:

```
# mmces service list -v
Enabled services: OBJ SMB NFS
 OBJ is running
    OBJ:openstack-swift-object-updater        is running
    OBJ:openstack-swift-object-expirer        is running
    OBJ:ibmobjectizer                         is running
    OBJ:openstack-swift-object-auditor        is running
    OBJ:openstack-swift-object                is running
    OBJ:openstack-swift-account               is running
    OBJ:openstack-swift-container             is running
    OBJ:memcached                             is running
    OBJ:openstack-swift-proxy                 is running
    OBJ:openstack-swift-object-replicator     is running
    OBJ:openstack-swift-account-reaper        is running
    OBJ:openstack-swift-account-auditor       is running
    OBJ:openstack-swift-container-auditor     is running
    OBJ:openstack-swift-container-updater     is running
    OBJ:openstack-swift-account-replicator    is running
    OBJ:openstack-swift-container-replicator  is running
    OBJ:openstack-swift-object-sof            is running
    OBJ:postgresql-obj                        is running
    OBJ:httpd (keystone)                      is running
 SMB is running
 NFS is running
```

For complete usage information, see *mmces command* in *IBM Spectrum Scale: Command and Programming Reference*.

Every object protocol node can access every virtual device in the shared file system, and some OpenStack Swift object services can be optimized to take advantage of this by running from a single Object protocol node.

Even though objects are not replicated by OpenStack Swift, the **swift-object-replicator** runs to periodically clean up tombstone files from deleted objects. It is run on a single Object protocol node and manages cleanup for all of the virtual devices.

The **swift-object-updater** is responsible for updating container listings with objects that were not successfully added to the container when they were initially created, updated, or deleted. Like the object replicator, it is run on a single object protocol node.

The following table shows each of the object services and the set of object protocol nodes on which they need to be executed.

Table 20. Object services and object protocol nodes

Object service	GPFS protocol node
ibmobjectizer	object-singleton_node [1]
openstack-swift-account	All
openstack-swift-account-auditor	object_singleton_node
openstack-swift-account-reaper	All
openstack-swift-account-replicator	All
openstack-swift-container	All
openstack-swift-container-auditor	object_singleton_node
openstack-swift-container-updater	object_singleton_node
openstack-swift-container-replicator	All
openstack-swift-object	All
openstack-swift-object-auditor	object_singleton_node [2]
openstack-swift-object-replicator	All
openstack-swift-object-sof	All [1]
openstack-swift-object-updater	object_singleton_node
openstack-swift-object-expirer	object_singleton_node
openstack-swift-proxy	All
memcached	All
openstack-keystone	All [3, 4]
postgresql-obj	object_database_node [3]

[1] If unified file and object access is enabled.
[2] If multi-region object deployment is enabled.
[3] If local OpenStack Keystone Identity Service is configured.
[4] Updated to httpd (keystone) on all nodes, if using local authentication.

Understanding the mapping of OpenStack commands to IBM Spectrum Scale administrator commands

Use this information to map OpenStack commands to IBM Spectrum Scale administrator commands.

In IBM Spectrum Scale, for Object storage, several OpenStack commands have been replaced with IBM Spectrum Scale commands for easy maintenance. This section identifies those commands.

1. Ring Building

   The swift-ring-builder command should only be used to view the object, container, and account ring on any IBM Spectrum Scale protocol node. The user should not directly execute any commands that modify the ring. All ring maintenance operations are handled automatically by the CES infrastructure.

   For example, when a new CES IP address is added to the configuration, all rings are automatically updated to distribute Swift virtual devices evenly across CES IP addresses.

   The master copy of each ring builder file is kept in the IBM Spectrum Scale Cluster Configuration Repository (CCR). Changes made locally to the ring files will be overwritten with the master copy when monitoring detects a difference between the ring file in CCR and the file in /etc/swift.

2. Configuration Changes

The openstack-config command should not be used to update any of the configuration files consumed by IBM Spectrum Scale for Object storage. Furthermore, you should not edit these files directly, but instead modify them using the **mmobj config change** command.

The master copy of object and related configuration files are kept in the IBM Spectrum Scale CCR. Changes made locally to these config files will be overwritten with the master copy when monitoring detects a difference between the configuration file in CCR and the file in `/etc/swift` or `/etc/keystone`.

Changing Object configuration values

Use the following information to change the Object configuration values in the Cluster Configuration Repository (CCR).

You can manage the Object configuration data in the Cluster Configuration Repository (CCR). When an object configuration is changed, callbacks on each protocol node updates that node with the change and restart one or more Object services if necessary.

To change the Object configuration, use the **mmobj** command so that the change is made in the CCR and propagated correctly across the Swift cluster.

For more details, see the *mmobj command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Changing the object base configuration to enable S3 API

IBM Spectrum Scale uses Swift3 Middleware for OpenStack Swift, allowing access to IBM Spectrum Scale using Amazon Simple Storage Service (S3) API.

Perform the following steps if S3 API was not enabled as part of the object base configuration:
1. To enable S3 API, run the following command:

   ```
   mmobj s3 enable
   ```
   The system enables S3 API.
2. To verify that S3 API has been enabled, run the following command:

   ```
   mmobj s3 list
   ```
3. To disable S3 API, run the following command:

   ```
   mmobj s3 disable
   ```
   The system disables S3 API.
4. To verify that S3 API has been disabled, run the following command:

   ```
   mmobj s3 list
   ```

Configuring OpenStack EC2 credentials

The credentials that are used on the Amazon S3 and Elastic Compute Cloud (EC2) APIs are different from the credentials that are used by the OpenStack API. As a result, you must generate these special credentials to use them when accessing the IBM Spectrum Scale OpenStack services.

The credentials are created by the OpenStackClient, a command-line client for OpenStack, that allows the creation and use of access/secret pairs for a user/project pair. This requires the operators to create the access/secret for each user/project pair.
1. Source openrc with the admin credentials.
2. Create EC2 credential by running this command for user-defined blob as a credential:

   ```
   openstack credential create --type ec2 --project <project> <user> '{"access": "<aws_access_key>", "secret": "<aws_secret_key>"}'
   ```

Note: Ensure to use Keystone UUIDs rather than names if duplicate user/project names exist across domains. Additionally, the admin users should be able to list and delete access/secrets for a specific user/project.

3. View all EC2 credentials by running this command:

```
openstack credential list
openstack credential show <credential-id>
```

4. You can change your Access Key ID and Secret Access Key if necessary.

   It is recommended to have regular rotation of these keys and switching applications to use the new pair.

   Change the EC2 credentials by running this command:

```
openstack credential set –type ec2 –data '{"access": <access>, "secret": <secret>}' --project <project> <credential-id>
```

5. Delete the EC2 credentials by running this command:

```
openstack credential delete <credential_id>
```

   The following example shows the creation of EC2 credentials using the admin project and the admin user IDs:

   Where openrc contains:

   export OS_AUTH_URL="http://127.0.0.1:35357/v3"

   export OS_IDENTITY_API_VERSION=3

   export OS_AUTH_VERSION=3 export

   OS_USERNAME="admin"

   export OS_PASSWORD="Passw0rd"

   export OS_USER_DOMAIN_NAME=Default

   export OS_PROJECT_NAME=admin

   export OS_PROJECT_DOMAIN_NAME=Default

   source openrc

```
openstack credential create --type ec2 --project admin admin '{"access": "022AB06E7MXBSH9DHMO2", "secret": "pWcuIUX4JEDGM/LtmEENI/aVmYvHNif5zB+d9+ct"}'
```

You are now ready to connect to the IBM Spectrum Scale Object store using the Amazon S3 API. You can connect with any S3-enabled client.

Managing OpenStack access control lists using S3 API

The following topic lists the permissions and the known limitations of S3 API.

IBM Spectrum Scale supports S3 access control lists (ACLs) on buckets and objects. These S3 ACLs are stored separately from the ACLs set through the Swift API and the ACLs stored in the file system (NFSv4 or POSIX). For information on how to set and query ACLs through S3 API, see the Amazon S3 documentation.

If S3 API is enabled, the default value of `s3_acl` in the `proxy-server.conf` file is `true`. S3 API uses its own metadata for ACL, such as X-Container-Sysmeta-Swift3-Acl, to achieve the best S3 compatibility. However, if S3 API is set to `false`, S3 API tries Swift ACLs, such as X-Container-Read, initially instead of S3 ACLs.

For a user to use S3 API in IBM Spectrum Scale, the user must have a role defined for the swift project. Any role suffices, because for S3 API there is no difference between the `SwiftOperator` role or others.

The owner of a resource is implicitly granted **FULL_CONTROL** instead of just **READ_ACP** and **WRITE_ACP**. This is not a security issue because with **WRITE_ACP**, the owners can grant themselves **FULL_CONTROL** access.

The following table lists the required permissions for S3 operations.

S3 operation	Required permission
PUT object	WRITE permission on bucket or as bucket owner
HEAD object	READ permission on object or as object owner
GET object	READ permission on object or as object owner
DELETE object	WRITE permission on bucket or as bucket owner
Get object ACL (GET on ACL subresource)	READ_ACP permission on object or as object owner
Set object ACL (PUT on ACL subresource)	WRITE_ACP permission on object or as object owner
Create bucket (PUT)	Any user with a role on the project can create a bucket.
HEAD bucket	READ permission on bucket or as bucket owner
GET bucket	READ permission on bucket or as bucket owner
DELETE bucket	bucket owner
Get bucket ACL (GET on ACL subresource)	READ_ACP permission on bucket or as bucket owner
Set bucket ACL (PUT on ACL subresource)	WRITE_ACP permission on bucket or as bucket owner

Known limitations for S3 API support

- Unauthorized S3 requests are not supported. S3 requests do not contain a reference to the account, and the object server derives the account information from the authorization information. This is not possible for unauthorized requests.
- Specifying S3 ACL grantees by email is not supported.
- Grantees in ACL are not validated. Therefore, any name can be used, even users that do not exist.
- The S3 ACLs are not supported in the Objects page of the IBM Spectrum Scale GUI.
- Container or objects created using the swift API are not accessible through S3 API when the `allow_no_owner` configuration flag is set to `false` in `proxy-server.conf`. To change this setting, you can use the following command:

```
mmobj config change --ccrfile proxy-server.conf --section filter:swift3
--property allow_no_owner --value true
```

  The default value of the `allow_no_owner` configuration flag is `true`.
- The POST operation to update metadata has not been implemented.

Managing object capabilities

You can manage the object capabilities using the following commands.

For an overview of object capabilities, see *Object capabilities* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- To list all object capabilities available cluster wide, use the **mmobj config list** command as follows:

```
mmobj config list --ccrfile spectrum-scale-object.conf --section capabilities
```

  The system displays output similar to the following:

```
file-access-enabled:   true
multi-region-enabled:  true
s3-enabled:            false
```

  You can also list specific object capabilities using the **mmobj config list** command as follows:

```
mmobj config list --ccrfile spectrum-scale-object.conf --section capabilities
--property file-access-enabled
mmobj config list --ccrfile spectrum-scale-object.conf --section capabilities
--property multi-region-enabled
mmobj config list --ccrfile spectrum-scale-object.conf --section capabilities
--property s3-enabled
```

- Use the **mmobj** command to enable object capabilities:
  - To enable or disable the file-access object capability, use the **mmobj file-access enable** or the **mmobj file-access disable** command.
  - To enable or disable the multiregion object capability, use the **mmobj multiregion enable** or the **mmobj multiregion disable** command.
  - To enable or disable the s3 object capability, use the **mmobj s3 enable** or the **mmobj s3 disable** command.

Managing object versioning

See the following topics to enable and disable object versioning.

Enabling object versioning

Perform the following steps to enable object versioning.

1. Alter the `proxy-server` pipeline by running the following command:

   ```
   mmobj config list --ccrfile proxy-server.conf --section
   pipeline:main
   ```

   The system displays the following output:

   ```
   [pipeline:main]
   pipeline = healthcheck cache formpost tempurl s3token authtoken
   keystoneauth container-quotas account-quotas staticweb bulk slo
   dlo versioned_writes proxy-server
   ```

2. Add `filter:versioned_writes`:

   ```
   mmobj config list --ccrfile proxy-server.conf --section
   filter:versioned_writes
   ```

   The system displays the following output:

   ```
   [filter:versioned_writes]
   use = egg:swift#versioned_writes
   allow_versioned_writes = true
   ```

Disabling object versioning

Perform the following steps to disable object versioning.

To disable object versioning across the cluster, run the following command:

```
# mmobj config change --ccrfile proxy-server.conf --section
'filter:versioned_writes' --property allow_versioned_writes --value false
```

The system displays the following output:

```
[filter:versioned_writes]
use = egg:swift#versioned_writes
allow_versioned_writes = false
```

Creating a version of an object: Example

The following example can be used to understand how to create object versions.

1. Create a container with the **X-Versions-Location** header or add the header to an existing container. In this example, *version_container* is the container that holds old versions of objects and *container1* is a new or existing container for which object versioning is to be enabled.

   swift post -H "X-Versions-Location: version_container" container1

2. Run **swift stat** on *container1* to check that **X-Versions-Location** header is applied:

```
swift stat container1
              Account: AUTH_f92886c4e3a347a18c29bae581b36788
            Container: container1
              Objects: 0
                Bytes: 0
             Read ACL:
            Write ACL:
              Sync To:
             Sync Key:
        Accept-Ranges: bytes
      X-Storage-Policy: Policy-0
           Connection: keep-alive
          X-Timestamp: 1468226043.18746
           X-Trans-Id: tx8d17476a914a40d781b0a-0057835a01
         Content-Type: text/plain; charset=utf-8
    X-Versions-Location: version_container
```

3. If *version_container* does not exist, create a new container:

   swift post version_container

4. Run **swift list** at the account level to check that both containers are created successfully:

   ```
   swift list
   container1
   version_container
   ```

5. Upload an object to *container1*:

   swift upload container1 ImageA.jpg

6. Upload the second version of the object:

   swift upload container1 ImageA.jpg

7. Upload the third version of the object:

   swift upload container1 ImageA.jpg

8. Run **swift list** on the container to view the stored object:

   ```
   swift list container1
       ImageA.jpg
   ```

   Note: The *container1* container contains only the latest version of the objects. The older versions of object are stored in *version_container*.

9. Run **swift list** on *version_container* to see the older versions of the object:

   ```
   swift list version_container
    00aImageA.jpg/1468227497.47123
    00aImageA.jpg/1468227509.48065
   ```

10. To delete the latest version of the object, perform the DELETE operation on the object:

    ```
    swift delete container1 ImageA.jpg
    ImageA.jpg
    (deleted latest/third version)

    # swift list container1
    ImageA.jpg
    (Second version is now the latest version)

    # swift list version_container
    00aImageA.jpg/1468227497.47123
    (Initial version of the object)
    ```

Mapping of storage policies to filesets

For every storage policy created using the **mmobj policy create** command, one fileset is created or reused.

After a storage policy is created, you can specify that storage policy while creating new containers to associate that storage policy with those containers. When objects are uploaded into a container, they are

stored in the fileset that is associated with the container's storage policy. For every new storage policy, a new object ring is created. The ring defines where objects are located and also defines multi-region replication settings.

The name of the fileset can be specified optionally as an argument of the **mmobj policy create** command. An existing fileset can be used only if it is not being used for an existing storage policy.

If even one of these prerequisites is missing, the **mmobj policy create** command fails. Otherwise, the fileset is used and the softlinks for the devices that are given to the ring builder point to it. If no fileset name is specified with the **mmobj policy create** command, a fileset is created using the policy name as a part of the fileset name with the prefix obj_.

For example, if a storage policy with name Test is created and no fileset is specified, a fileset with the name obj_Test is created and is linked to the base file system for object:

```
<object base filesystem mount point>/obj_Test/<n virt. Devices>
```

Attention: For any fileset that is created, its junction path is linked under the file system. The junction path should not be changed for a fileset that is used for a storage policy. If it is changed, data might be lost or it might get corrupted.

To enable swift to work with the fileset, softlinks under the given devices path in `object-server.conf` are created:

```
<devices path in object-server.conf>/<n virt. Devices>
<object base filesystem mount point>/obj_Test/<n virt. Devices>
```

Administering storage policies for object storage

Use the following information to create, list, and change storage policies for object storage.

Before creating a storage policy with the file-access (unified file and object access) function enabled, the file-access object capability must be enabled. For more information, see "Managing object capabilities" on page 242 and *Object capabilities* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- To create a new storage policy with the unified file and object access feature enabled, run the following command:

  ```
  mmobj policy create sof-policy --enable-file-access
  ```

  The system displays output similar to this:

  ```
  [I] Getting latest configuration from ccr
  [I] Creating fileset /dev/gpfs0:obj_sof-policy
  [I] Creating new unique index and build the object rings
  [I] Updating the configuration
  [I] Uploading the changed configuration
  ```

- To list storage policies for object storage with details of functions available with those storage policies, run the following command:

  ```
  mmobj policy list --verbose
  ```

  The system displays output similar to the following example:

```
Index        Name         Deprecated Fileset  Fileset Path                  Functions              Function Details  File System
-------------------------------------------------------------------------------------------------------------------------------
0            SwiftDefault  object_fileset      /ibm/cesSharedRoot/object_fileset                                      cesSharedRoot
11751509160  sof-policy    obj_sof-policy      /ibm/cesSharedRoot/obj_sof-policy  file-and-object-access  regions="1"  cesSharedRoot
11751509230  mysofpolicy   obj_mysofpolicy     /ibm/cesSharedRoot/obj_mysofpolicy file-and-object-access  regions="1"  cesSharedRoot
```

- To change a storage policy for object storage, run the following command:

  ```
  mmobj policy change
  ```

- To change the default storage policy, run the following command:

  ```
  mmobj policy change sof-policy --default
  ```

  The system displays `sof-policy` as the default storage policy.

For more information about the **mmobj policy** command, see *mmobj command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Creating storage policy for object compression

Use the following information to create a storage policy with the compression function enabled and to create a storage policy with the compression schedule defined.

- To create a storage policy with the compression function enabled, use the `--enable-compression` option with the **mmobj policy create** command as follows:

  ```
  mmobj policy create CompressionTest --enable-compression --compression-schedule "MM:HH:dd:ww"
  ```

- To create a storage policy with the compression function enabled and a compression schedule defined, use the `--enable-compression` and the `--compression-schedule` options with the **mmobj policy create** command as follows:

  ```
  mmobj policy create CompressionTest --enable-compression --compression-schedule "MM:HH:dd:ww"
  ```

  ```
  where
  MM = 0-59 minutes
  HH = 0-23 hours
  dd = 1-31 day of month
  ww = 0-7 (0=Sun, 7=Sun) day of week
  ```

  - Use * for specifying every instance of a unit. For example, dd = * means that the job is scheduled to run every day.
  - Comma separated lists are allowed. For example, dd = 1,3,5 means that the job is scheduled to run on every 1st, 3rd, 5th of a month.
  - If ww and dd both are specified, the union is used.
  - Specifying a range using - is not supported.
  - Empty values are allowed for dd and ww. If empty, dd and or ww are not considered.
  - Empty values for mm and hh are treaded as *.

  In the following example, the compression job has been scheduled to run at 23.50 every day:

  ```
  mmobj policy create CompressionTest --enable-compression --compression-schedule "50:23:*:*"
  ```

  Every object stored using a storage policy that has compression enabled is compressed according to the specified schedule. There is no need to decompress an object in advance of a get request or any other object request. IBM Spectrum Scale automatically returns the decompressed object.

  Note: The download performance of objects in a compressed container is reduced compared to the download performance of objects in a non-compressed container.

  Note: The compression function enables the file system compression over the object file set. The same compression functionality and restrictions apply to object compression and file compression.

Related concepts:

➡ File compression
You can compress or decompress files either with the **mmchattr** command or with the **mmapplypolicy** command with a **MIGRATE** rule. You can do the compression or decompression synchronously or defer it until a later call to **mmrestripefile** or **mmrestripefs**.

Creating storage policy for object encryption

Use the following information to create a storage policy for encryption.

To create a storage policy with the encryption function enabled, use the **mmobj policy create** command as follows:

```
mmobj policy create PolicyName -f FilesetName -i MaxNumInodes
--enable-encryption --encryption-keyfile EncryptionKeyFileName
--force-rule-append
```

where:

PolicyName
> The name of the storage policy to be created.

FilesetName
> The fileset name that the created storage policy must use. Optional.

FilesystemName
> The file system name where the fileset resides. Optional.

MaxNumInodes
> The inode limit for the new inode space. Optional.

--enable-ecryption
> Enables an encryption policy.

EncryptionKeyFileName
> The fully qualified path of the encryption key file.

--force-rule-append
> Adds and establishes the rule if other rules already exist. Optional.

The **--force-rule-append** determines whether to establish the GPFS policy rules:

* If **--force-rule-append** is not set:
  - During create policy, the command checks whether a GPFS policy rule is already established.
  - If so, the new encryption rule is not established but is displayed.
  - Otherwise, the new encryption rule is established and is displayed.
* If it is set:
  - During create policy, the command checks whether a GPFS policy rule is already established.
  - If so, the new encryption rule is added to the already established rules and the GPFS policy for the file system is updated. The new encryption rule is displayed..
  - Otherwise, the new encryption rule is established and is displayed.

During command execution the encryption policy and rule is created. A GPFS policy rule file is created and used to establish the policy rule.
Policy Rule File:

* filename = /var/mmfs/ces/policyencryption.rule

Note: The filename is autogenerated.
After the encryption policy is created, depending on the presence or absence of the **--force-rule-append** parameter, the command displays the new encryption policy.
If an error on creating the encryption part occurs, the local cleanup function is called to remove the just created fileset and exit the CLI mmobj policy create script. The existing rules and policies are not changed.

Note: The encryption function enables the file system encryption over the object file set. The same encryption functionality and restrictions apply to object encryption and file encryption.

Adding a region in a multi-region object deployment

Perform the following steps to add a region in a multi-region object deployment environment.

In the command examples, Europe is the first region and Asia is the second region.

1. Export the information of the first region to a file by using the **mmobj multiregion export** command. For example:

   ```
   [europe]# mmobj multiregion export --region-file /tmp/multiregion_europe.dat
   ```

2. Copy the file manually to the second region.

   For example:

   ```
   [europe]# scp /tmp/multiregion_europe.dat asia:/tmp
   ```

3. From the second region, join the multi-region environment as follows:

   a. Use the file generated in the first region while deploying object on the second region by using the **mmobj swift base** command.

      For example:

      ```
      [asia]# mmobj swift base -g /mnt/gpfs0 --cluster-hostname gpfs-asia --admin-password xxxx
      -i 100000 --admin-user admin \
        --enable-multi-region  --remote-keystone-url  http://gpfs-asia:35357/v3
       --join-region-file /tmp/multiregion_europe.dat \
              --region-number 2 --configure-remote-keystone
      ```

      This step installs the object protocol in the 2nd region and joins the 1st region. Additional devices are added to the primary ring files for this region.

4. Export the ring file data of the second region.

   For example:

   ```
   [asia]# mmobj multiregion export --region-file /tmp/multiregion_asia.dat
   ```

5. Copy the file manually to the first region.

   For example:

   ```
   [asia]# scp /tmp/multiregion_asia.dat europe:/tmp
   ```

6. In the first region, update the local ring files with the configuration of the second region.

   For example:

   ```
   [europe]# mmobj multiregion import --region-file /tmp/multiregion_asia.dat
   ```

   This step reads in the ring files which are updated with the information of the second region. This update ensures that the data of the second region contains a new region and therefore replaces the associated ring files in the first region with the ones from the second region.

Note:

Now the two clusters have been synced together and can be used as a multi-region cluster. Objects can be uploaded and downloaded from either region. If the installation of the second region specified the **--configure-remote-keystone** flag, a region-specific endpoint for the object-store service for the 2nd region is created in Keystone.

The regions need to be synced in the future any time region-related information changes. This includes changes in the set of CES IP addresses (added or removed) or if storage policies were created or deleted within a region. Changes that affect the `swift.conf` file or ring files need to be synced to all regions. For example, adding additional CES addresses to a region causes the ring files to be rebuilt.

7. In the second region, add CES addresses and update other clusters.

   For example:

   ```
   [asia]# mmces address add --ces-ip asia9
   ```

   This steps adds an address to the CES IP pool. This also triggers a ring rebuild which changes the IP-to-device mapping in the ring files.

8. Export the ring data so the other clusters in the region can be updated with the new IPs from the second region.

   For example:

   ```
   [asia]# mmobj multiregion export --region-file /tmp/multiregion_asia.dat
   ```

9. Copy the file manually to the first region.

   For example:

   ```
   [asia]# scp /tmp/multiregion_asia.dat europe:/tmp
   ```

10. In the first region, update with changes for the new second region address in the ring.

    For example:

    `[europe]# mmobj multiregion import --region-file /tmp/multiregion_asia.dat`

    This step imports the changes from the second region. When this is complete, a checksum is displayed which can be used to determine when regions are synchronized together. By comparing it to the one printed when the region data was exported, you can determine that the regions are synchronized when they match. In some cases, the checksums do not match after import. This is typically due to some local configuration changes on this cluster which are not yet synced to the other regions. If the checksums do not match, then this region's configuration needs to be exported and imported into the other region to sync them.

Administering a multi-region object deployment environment

Use the following information to administer a multi-region object deployment environment.

A multi-region environment consists of several independent storage clusters linked together to provide unified object access. Configuration changes in one cluster which affect the multi-region environment are not automatically distributed to all clusters. The cluster which made the configuration change must export the relevant multi-region data and then the other regions must import that data to sync the multi-region configuration. Changes which affect multi-region are:

- Changes to the CES IP pool, such as adding or deleting addresses, which affect the ring layout.
- Changes to the object services ports used for the account, container, and object servers (ports 6200-6202).
- Creation, deletion, or modification of storage policies.
- Changes to the `swift.conf` configuration file

Use the following commands to manage the configuration of the multi-region environment:

- To export the data for the current region so that it can be integrated into other regions, use the following command. The *RegionData* file created can be used to update other regions:

  `mmobj multiregion export --region-file RegionData`

  The *RegionData* file is created and it contains the updated multi-region information.

- To import the multi-region data to sync the configuration, use the following command. The *RegionData* must be the file created from the **mmobj multiregion export** command:

  `mmobj multiregion import --region-file RegionData`

As part of the export/import commands, a region checksum is printed. This checksum can be used to ensure that the regions are in sync. If the checksums match, then the multi-region configuration of the clusters match. In some cases, the checksums do not match after import. This is because the cluster performing the import had local configuration changes which had not been synced with the other regions. For example, a storage policy was created but the multi-region configuration was not synced with the other regions. When this happens the import command prints a message that the regions are not fully in sync because of the local configuration and that the region data must be exported and imported to the other regions. Once all regions have matching checksums, the multi-region environment is in sync.

An existing region can be completely removed from the multi-region environment. This action permanently removes the region configuration, and the associated cluster cannot rejoin the multi-region environment.

The cluster of the removed region needs to disable object services since it will not be usable as a standalone object deployment.

- To remove a previously defined region from the configuration, use the following command:

  `mmobj multiregion remove --remove-region-number RegionNumber`

The remove command must be run from a different region than the one being removed. The cluster associated with the removed region must cleanup object services as appropriate with the `mmces service disable OBJ -a` command to uninstall object services.

- You can display the current multi-region information using the following command:
  ```
  mmobj multiregion list
  ```

Unified file and object access in IBM Spectrum Scale

Unified file and object access allows use cases where you can access data using object as well as file interfaces. Use the following information to manage unified file and object access including identity management modes for unified file and object access, authentication for unified file and object access, and objectization service schedule.

Important: In a unified file and object access environment, object ACLs apply only to accesses through the object interface and file ACLs apply only to accesses through the file interface.

For example: If user Bob ingests a file from the SMB interface and user Alice does not have access to that file from the SMB interface, it does not mean that Alice does not have access to the file from the object interface. The access rights of Alice for that file or object from the object interface depends on the ACL defined for Alice on the container in which that file or object resides.

Enabling object access to existing filesets

Learn to enable object access to files stored in an existing fileset.

"Enabling access with --update-listing"

Note: Before you enable object access for the existing filesets, ensure that SELinux is in the Permissive or Disabled mode.

Enabling access with --update-listing

This set of examples uses the following resources:

- The account name is `admin`.
- The policy name is `sof_policy`.
- The file system name is `gpfs1`.
- The fileset name is `legacy_fset1`.
- The fileset junction path is `/gpfs1/legacy_fset1`, which contains the following files:
    ```
    existingfile1
    existingdir/existingfile2
    ```
- The container name is `cont1`.

You can do the following operations. All the commands are on one line:

1. The following command enables object access to the fileset and updates the container listing with the existing files:
   ```
   mmobj file-access link-fileset --sourcefset-path /gpfs1/legacy_fset1
       --account-name admin --container-name cont1 --fileaccess-policy-name sof_policy
       --update-listing
   ```
   The command also creates the soft link `gpfs1-legacy_fset1`. The link name is constructed according to the following format: *<file_system_name>-<fileset_name>*.

2. Both of the following commands upload an object `newobj` to the linked fileset path `/gpfs1/legacy_fset1`. Both commands use the soft link `gpfs1-legacy_fset1` that is created in the preceding example. You can use either method:

   - The following example uses the **swift** utility:
     ```
     swift upload -H "X-Storage-Policy: sof_policy" cont1 'gpfs1-legacy_fset1/newobj'
     ```
   - The following example uses the **curl** utility:
     ```
     curl -X PUT -T newobj -H "X-Storage-Policy: sof_policy"
     http://specscaleswift.example.com:8080/v1/AUTH_cd1a29013b6842939a959dbda95835df/cont1/gpfs1-legacy_fset1/newobj
     ```

   The following command creates a new directory `newdir` and uploads the object `newobj1` to it:
   ```
   swift upload -H "X-Storage-Policy: sof_policy" cont1 'gpfs1-legacy_fset1/newdir/newobj1'
   ```

3. The following command lists the contents of the container `cont1`:
   ```
   swift list cont1
   ```

   The command displays the following output:
   ```
   gpfs1-legacy_fset1/newdir/newobj1
   gpfs1-legacy_fset1/newobj
   gpfs1-legacy_fset1/existingfile1
   gpfs1-legacy_fset1/existingdir/existingfile2
   ```

4. The following command downloads the file `newobj`:
   ```
   swift download cont1 'gpfs1-legacy_fset1/newobj'
   ```

Enabling access without --update-listing

This set of examples uses the following resources:
- The account name is `admin`.
- The policy name is `sof_policy`.
- The file system name is `gpfs1`.
- The fileset name is `legacy_fset2`.
- The fileset junction path is `/gpfs1/legacy_fset2`, which contains the following file:
    ```
    existingfile2
    ```
- The container name is `cont2`.

You can do the following operations. All the commands are on one line:
1. The following command enables object access to the fileset and creates the link `gpfs1-legacy_fset2` but does not update the container listing with existing files:
   ```
   mmobj file-access link-fileset --sourcefset-path /gpfs1/legacy_fset2
       --account-name admin --container-name cont2 --fileaccess-policy-name sof_policy
   ```
2. The following command uploads the object `newobj` to the linked fileset path `/gpfs1/legacy_fset2`:
   ```
   swift upload -H "X-Storage-Policy: sof_policy" cont2 'gpfs1-legacy_fset2/newobj'
   ```
3. The following command lists the contents of the container `cont2`:
   ```
   swift list cont2
   ```

   The command displays the following output.
   ```
   gpfs1-legacy_fset2/newobj
   ```

   Note that the command displays only the objects that are added to the fileset, either by uploading the object or by specifying the **--update-listing** parameter with **mmobj --file-access**. Here the only such object is `newobj`. The command does not list the existing file `existingfile2`.

Enabling access with a fileset path from a different object file system

You can enable object access to an existing non-object fileset path where the fileset path is derived from a different object file system. To do so, omit the **--update-listing** parameter. You can access the data with the utilities **swift** or **curl**. However, the container listing is not updated with the existing file entries and object metadata is not appended to the existing data.

Identity management modes for unified file and object access

The following section gives information about the two identity management modes for unified file and object access: local mode and unified mode. This section also describes how to configure these modes for a system.

Unified file and object access comprises the following two modes:
- **local_mode**: Separate identity between object and file (Default mode)
- **unified_mode**: Shared identity between object and file

The mode is represented by the **id_mgmt** configuration parameter in the `object-server-sof.conf` file:

```
id_mgmt = local_mode | unified_mode
```

You can change this parameter by using the **mmobj config change** command. For more information, see "Configuring authentication and setting identity management modes for unified file and object access" on page 262.

Note:
- Only one mode can be effective at a given time and it needs to be configured by the administrator for the entire system. **id_mgmt = local_mode** is the default setting.
- If you plan to use `unified_mode`, the authentication mechanism for file and object must be the same. If you set **id_mgmt** to `unified_mode` and the file authentication and object authentication are not common, then the ID resolution of the users will not work correctly. This will lead to either object not being created with 503 error* return code or object being created with improper user ID. Therefore, it is very important that the administrators ensure that a common authentication with appropriate ID mapping is configured for file and object.

  * If you are using swift client, instead of 503, you might get an error similar to the following:

  ```
  'put_object('container_name', 'object_name', ..) failure and no ability to reset contents for reupload.'
  ```

  For more information about validating the ID mapping, see *Validating shared authentication ID mapping* in *IBM Spectrum Scale: Administration Guide*

local_mode - separate identity between object and file

The following points should be considered when planning to use local_mode identity management.
- Use-case for unified file and object access in `local_mode`:
  - Data created from the object interface is available for application to run analytics using the file interface, where ownership of files is not essential.
  - Data created from the file interface is accessible from the object interface after objectization of those files.
- To address this use case, object authentication setup is independent of file authentication setup. Although, you can set up object and file authentication from a common authentication server in case of AD or LDAP.
- Objects created or updated using the object interface are owned by the `swift` user. Application processing the object data from file interface need the required file ACL to access the object data.
- Data updated from the file interface after objectization is available for object access.

- Containers created with a unified file and object access policy that are exposed as export points need appropriate ACLs set as needed by SMB, NFS, and POSIX.
- If the object already exists, existing ownership of the corresponding file is retained if `retain_owner` is set to `yes` in `object-server-sof.conf`. For more information, see "Configuration files for IBM Spectrum Scale for object storage" on page 272.
- Retaining ACL, extended attributes (xattrs), and Windows attributes (winattrs): If the object is created or updated over existing file then existing file ACL, xattrs, and winattrs are retained if `retain_acl`, `retain_xattr`, and `retain_winattr` are set to `yes` in `object-server-sof.conf`. For more information, see "Configuration files for IBM Spectrum Scale for object storage" on page 272.
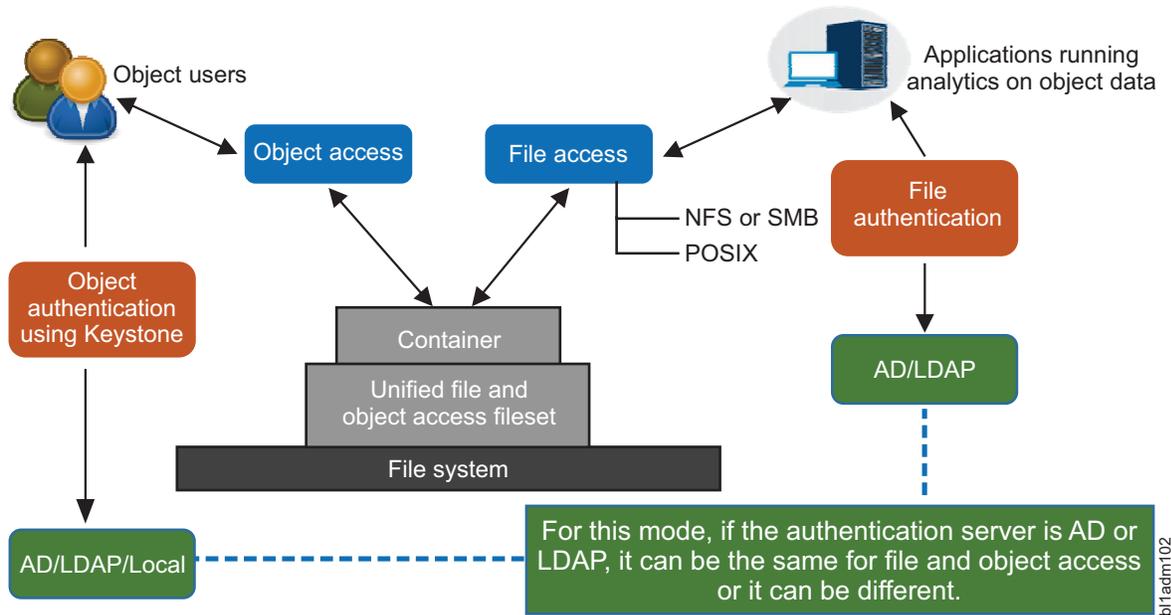
Figure 5. `local_mode` - separate identity between object and file

unified_mode - shared identity between object and file

The following points should be considered when using unified_mode identity management.

- Users from object and file are expected to be common and coming from the same directory service (only AD+RFC 2307 or LDAP).

    Note: If your deployment uses only SMB-based file interface for file access and file authentication is configured with Active Directory (AD) with Automatic ID mapping, unified file and object access can be used, assuming that object is configured with the same AD domain.
- Ownership: Object created from the object interface is owned by the user doing the object PUT operation.
- If the object already exists, existing ownership of the corresponding file is retained if `retain_owner` is set to `yes` in `object-server-sof.conf`. For more information, see "Configuration files for IBM Spectrum Scale for object storage" on page 272.
- Authorization: Object access follows the object ACL semantics and file access follows the file ACL semantics.
- Retaining ACL, extended attributes (xattrs), and Windows attributes (winattrs): If the object is created or updated over existing file then existing file ACL, xattrs, and winattrs are retained if `retain_acl`, `retain_xattr`, and `retain_winattr` are set to `yes` in `object-server-sof.conf`. For more information, see "Configuration files for IBM Spectrum Scale for object storage" on page 272.

- When a user does a PUT operation for an object over an existing object or does a PUT operation for a fresh object over a nested directory, no explicit file ACL is set for that user. This means that it is possible that in some cases, the user might not have access to that file from the file interface even though the user has access from the object interface. This is done to prevent changing of the file ACL from the object interface to maintain file ACL semantics. In these cases, if the user is required to have permission to access the file also, explicit file ACL permission need to be set from the file interface.

  For example: If user Bob performs a PUT operation for an object over an existing object (object maps to a file) owned by user Alice, Alice continues to own the file and there is no explicit file level ACL that is set for Bob for that file. Similarly, when Bob performs a PUT operation for a new object inside a subdirectory (already created by Alice), no explicit file ACL is set on the directory hierarchy for Bob. Bob does not have access to the object from the file interface unless there is an appropriate directory inheritance ACL that is set. To summarize, the object ingest does not change any file ACL and vice versa.

Table 21. Object input behavior in `unified_mode`.

Note: In the scenarios listed in the following table, the operations are being done by user **Bob** from the object interface. The instances owned by user Alice imply that the file or directory ownership maps to user **Alice** from the file side. Also, it is assumed that the `retain_owner`, `retain_acl`, `retain_xattr`, and `retain_winattr` parameters are set to `yes` in `object-server-sof.conf`.

Operation from SWIFT interface on object or container	Ownership result on corresponding file or directory		ACL, xattr, and winattr retention behavior on corresponding file or directory	
	File	**Directory**	**File**	**Directory**
Bob does a PUT operation for a new object	The ownership of the file is set to Bob	NA	Default GPFS ACLs are set	NA
Bob does a PUT operation for a new container	NA	The ownership of the directory is set to Bob	NA	Default GPFS ACLs are set
Bob does a PUT operation for an object that is already present and is owned by Alice	The ownership of the file continues to be with Alice. Bob is not given any file ACL explicitly.	No changes in the ownership of the parent directory	Existing ACL, file xattrs, and file winattrs are retained**	NA
Bob does a POST operation (update metadata) of existing object owned by Alice	The ownership of the file continues to be with Alice. Bob is not given any file ACL explicitly	NA	Existing ACL, file xattrs, and file winattrs are retained**	NA
Bob does a POST operation (update metadata or ACL) of existing container owned by Alice	NA	The ownership of the directory continues to be with Alice. Bob is not given any directory ACL	NA	NA
GET/DELETE/HEAD	No impact			

Note: **Unified file and object access retains the extended attributes (xattr), Windows attributes (winattrs) and ACL of the file if there is a PUT request from an object over an existing file. However, security or system namespace of extended attributes and other IBM Spectrum Scale extended attributes such as immutability, pcache, etc. are not retained. Swift metadata (`user.swift.metadata`) is also not retained and it is replaced according to object semantics which is the expected behavior.

Advantages of using `unified_mode`

IBM Spectrum Scale offers various features that leverage user identity (UIDs or GIDs). With `unified_mode`, you can use these features seamlessly across file and object interfaces.

- **Unified access to object data:** User can access object data using SMB or NFS exports using their AD or LDAP credentials.
- **Quota:** GPFS quota for users that work on UID or GID can be set such that they work for the file as well as object interface.

  Example: User A can have X quota on a unified access fileset assigned using GPFS quota commands which can hold true for all data created by the user from the file or the object interface.

  For more information, see the *Quota related considerations for `unified_mode`* section.

- **ILM:** Tiering of user specific data leveraging UID or GID.

  Example 1: The UID and GID file attributes can be used to create an ILM placement policy to place the files owned by the Gold customers in faster storage pools and retain the files in the pools even when the pool storage starts reaching the threshold. The UID and GID file attributes can also be used to create a migration ILM policy so that, when the pool reaches its storage threshold, all files older than 30 days are moved to a slower storage pool except the ones owned by the Gold customers.

  Example 2: After a user has left the organization, the UID of the user can be used to migrate the data and retain it on the archive tape for as long as defined as defined by the ILM policies.

- **Backup:** Backup of user specific data leveraging UID or GID.

  Example: UID and GID file attributes in the policy rules that are defined for the mmbackup command can be used to regularly back up the data of selective users.
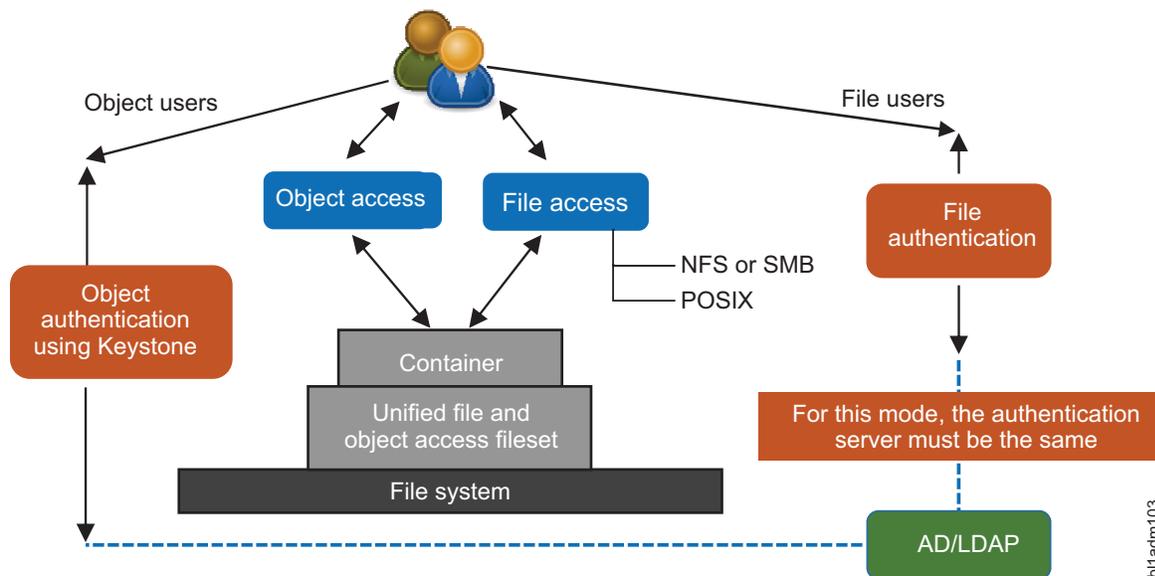
Figure 6. `unified_mode` - unified identity between object and file

Quota related considerations for `unified_mode`

There are three types of quotas that need to be considered:

- Quota for a user set using file system commands for that fileset which is set using User ID or Group ID. This quota represents the size in bytes up to which the user can create data on a given fileset. This is tracked at the file system level.

- Container quota: This is the size in bytes or number of objects that can be stored in a container regardless of the user making the upload (PUT) request. For more information on container quotas, see OpenStack documentation of container quotas.
- Account quotas: This is the size in bytes or number of objects that can be stored in an account regardless of the user making the upload (PUT) request. For more information on account quotas, see OpenStack documentation of account quotas.

The fileset quotas and container level quotas as well as fileset quotas and account quotas are independent of each other.

In some cases, the fileset quota should be cumulative of all the containers' quotas hosted over it, though it is not mandatory. When both the quotas at the fileset level as well as at the container quota level are set, and if the fileset quota is reached, no more object data can be input on any of the containers hosted by that fileset, despite of the container quota not being reached. Hence, when you plan to use both file and object quotas, it is important to understand these details.

The objectization process does not take into account the container quota and the account quota. This means that there might be a scenario where a container can host more data than the container quota associated with it especially when the **ibmobjectizer** service has objectized files as objects.

For example, consider that:
- You want to have a total of 1 TB of data allocated for file and object access.
- You want each user to have an overall quota from the file as well as the object interface to be 10 GB.
- You have a pre-defined set of 100 containers which are enabled for object and file access (using the storage policy for object storage) and users access to different containers is dependent on the container ACLs.

In this case, quotas are set as follows:
1. Set the fileset quota associated with the file access policy to 1 TB.
2. Set the user quota on that fileset to 10 GB.
3. Set the container quota to the required level. However, setting it more than fileset quota cannot be honored until fileset quota is increased or unset.

In this example scenario, note that the object access will be restricted if either the user quota or the fileset quota is reached, even though the container quota is not reached.

Authentication in unified file and object access

The following section gives information about how file authentication and object authentication are configured for different identity management modes.

Authentication configuration in local_mode: separate identity between object and file

In this mode, all the objects created continue to be owned by the swift user, that is an administrator under whose context the object server runs on the system. Because in this mode there is no ID mapping of objects to user ID, object authentication can be configured to any supported authentication schemes and file authentication can continue to be configured to any supported authentication scheme.

For supported authentication schemes, see the *Authentication support matrix* table in the *Authentication considerations* topic in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Authentication configuration in unified_mode: shared identity between object and file

This mode allows objects and files to be owned by the users' UID and the corresponding GID that created them. This mode mandatorily requires both the object protocol and the file protocol to be configured with the same authentication scheme. The supported authentication schemes for the unified mode are:

- AD for Authentication + RFC 2307 for ID mapping
- LDAP for authentication as well as for ID mapping

Note: User-defined authentication is not supported with both the identity management modes.

Validating shared authentication ID mapping

Perform the following steps to validate shared authentication ID mapping.

1. List the authentication details on IBM Spectrum Scale by running the `mmuserauth service list` command. The system displays the output similar to the following:

```
FILE access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_SERVER_TLS false
ENABLE_KERBEROS false
USER_NAME cn=manager,dc=sonasldap,dc=com
SERVERS 9.118.37.234
NETBIOS_NAME deepakcluster
BASE_DN dc=sonasldap,dc=com
USER_DN dc=sonasldap,dc=com
GROUP_DN none
NETGROUP_DN none
USER_OBJECTCLASS posixAccount
GROUP_OBJECTCLASS posixGroup
USER_NAME_ATTRIB cn
USER_ID_ATTRIB uid
KERBEROS_SERVER none
KERBEROS_REALM none

OBJECT access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND false
ENABLE_SERVER_TLS false
ENABLE_KS_SSL false
USER_NAME cn=manager,dc=sonasldap,dc=com
SERVERS 9.118.37.234
BASE_DN dc=sonasldap,dc=com
USER_DN dc=sonasldap,dc=com
USER_OBJECTCLASS posixAccount
USER_NAME_ATTRIB cn
USER_ID_ATTRIB uid
USER_MAIL_ATTRIB mail
USER_FILTER none
ENABLE_KS_CASIGNING false
KS_ADMIN_USER userr
```

2. Ensure that the file authentication type and the object authentication type are the same. The valid values are AD and LDAP. The following are examples of file authentication and object authentication types:

```
FILE access configuration : LDAP
OBJECT access configuration : LDAP
```

With AD configuration, file authentication must be configured with Unix mapped domain. And the object authentication should also be configured with the same AD domain. This AD domain should be updated in the `object-server-sof.conf` configuration as: `ad_domain = <AD domain name>`

3. Configure the file authentication and the object authentication against the same server:

```
FILE : SERVERS 9.118.37.234
OBJECT : SERVERS 9.118.37.234
```

Note: If there are multiple domain controllers in AD, the values might not match. The administrator must ensure that the server is referring to same user authentication source.

4. Ensure that the object users are receiving the correct UIDs and GIDs from the authentication source. In the following example, the userr object user, is being used:

```
cat openrc
# Thu May 26 19:04:37 IST 2016
export OS_AUTH_URL="http://127.0.0.1:35357/v3"
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
export OS_USERNAME="userr"
export OS_PASSWORD=""
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_PROJECT_DOMAIN_NAME=Default
```

5. Ensure that the object user is correctly resolved on all the protocol nodes and the same UID and GID are listed. In the following example, the UID and GID of the userr object user has been listed:

```
id userr
uid=1101(userr) gid=1000(testgrp) groups=1000(testgrp),1002(testgrp2)
```

The objectizer process

The objectization process converts files ingested from the file interface on unified file and access enabled container path to be available from the object interface. The name of the service that does this is **ibmobjectizer**.

When new files are added from the file interface, they need to be visible to the Swift database to show correct container listing and container or account statistics.

The **ibmobjectizer** service ensures synchronization between the file metadata and the object metadata at predefined time interval that ensures accurate container and account listing. The **ibmobjectizer** service identifies new files added from the file interface and adds the Swift system metadata to them so that they are objectized. The **ibmobjectizer** service then determines its container and account databases and adds a new object entry to those. It also identifies files deleted from file interface and deletes their corresponding entries from container and account databases.

This is particularly useful in setups where data is ingested using legacy file interface based devices such as medical and scientific devices and it needs to be stored and accessed over cloud using the object interface.

The **ibmobjectizer** service is a singleton and it is started when object is enabled and the file-access object capability is set. However, the **ibmobjectizer** service starts objectization only when there are containers with unified file and object access storage policies configured and the file-access object capability is set.

For use cases in which objects are always ingested using the object interface and the file interface is used only for reading them, the **ibmobjectizer** service is not needed and can be disabled using the **mmobj file-access** command. For more information, see "Starting and stopping the ibmobjectizer service" on page 261.

To identify the node on which the **ibmobjectizer** service is running, use the `mmces service list --verbose` command.

For example, if you have a cluster that has gpfsnode3 as the object singleton node, run the following command:

```
mmces service list --verbose -a | grep ibmobjectizer
```

The system displays the following output:

```
gpfsnode3:        OBJ:ibmobjectizer                       is running
```

Attention: If object services on the singleton node are stopped by the administrator manually, objectization is stopped across the cluster. Therefore, manually stopping services on a singleton node must be planned carefully after understanding its impact.

For information on limitations on the objectizer process, see "Limitations of unified file and object access" on page 268.

Related concepts:

"Understanding and managing Object services" on page 237
Use the following information to manage services related to IBM Spectrum Scale for object storage.

Related tasks:

"Setting up the objectizer service interval" on page 261
Take the following steps to set up the objectizer service interval.

Related reference:

"Configuration files for IBM Spectrum Scale for object storage" on page 272
Use the following information to manage options in configuration files that are used for IBM Spectrum Scale for object storage including the unified file and object access feature. These configuration files are located in the /etc/swift directory.

File path in unified file and object access

One of the key advantages of unified file and object access is the placement and naming of objects when stored on the file system.

Unified file and object access stores objects following the same path hierarchy as the object's URL. In contrast, the default object implementation stores the object following the mapping given by the ring, and its final file path cannot be determined by the user easily. For example, an object with the following URL is stored by the two systems as follows:

- **Example object URL:** `https://swift.example.com/v1/acct/cont/obj`
- **Path in default object implementation:** `/ibm/gpfs0/object_fileset/o/z1device108/objects/7551/125/75fc66179f12dc513580a239e92c3125/75fc66179f12dc513580a239e92c3125.data`
- **Path in unified file and object access:** `/ibm/gpfs0/obj_sofpolicy1/s69931509221z1device1/AUTH_763476384728498323747/cont/obj`

In this example, it is assumed that the object is configured over the `/ibm/gpfs0` file system with the default object located on the `object_fileset` fileset and the unified file and object access data is located under the `obj_sofpolicy1` fileset. `s69931509221z1device1` is auto-generated based on the swift ring parameters and `AUTH_763476384728498323747` is auto-generated based on the account ID from keystone.

Attention: Do not unlink object filesets including the unified file and object access enabled filesets.

Determining the POSIX path of a unified file and object access enabled fileset

Use the following steps for determining the POSIX path of a unified file and object access enabled fileset.

1. List all storage policies for object.

```
mmobj policy list

Index       Name        Default Deprecated Fileset          Functions            File System
---------------------------------------------------------------------------------------------------
0           SwiftDefault yes                object_fileset
```

```
13031510160 sof-policy1                        obj_sof-policy1    file-and-object-access  fs0
13031510260 CompressionTest        yes         obj_CompressionTest compression           fs0
13031510290 CompressionDebug       yes         obj_CompressionDebug compression          fs0
13031511020 CompressionNew                     obj_CompressionNew  compression           fs0
```

2. In the fs0 file system, note the index and fileset name for the policy you are interested in and use the **mmlsfileset** command to determine the junction point.

```
mmlsfileset fs0 | grep obj_sof-policy1
obj_sof-policy1            Linked    /ibm/fs0/obj_sof-policy1
```

The swift ring builder creates a single virtual device for unified file and object access policies, named with storage policy index number, which is also the region number, starting with s and appended with z1device1.

```
s13031510160z1device1
```

3. List the swift projects and identify the one you are interested in working with.

```
~/openrc
openstack project list
+----------------------------------+---------+
| ID                               | Name    |
+----------------------------------+---------+
| 73282e8bca894819a3cf19017848ce6b | admin   |
| 1f78f58572f746c39247a27c1e0e1488 | service |
+----------------------------------+---------+
```

4. Construct the account name by appending the project ID with AUTH_ or substitute the correct project prefix if you have customized this. For the admin project, use:

```
AUTH_73282e8bca894819a3cf19017848ce6b
```

The full path to the unified file and object access containers is the concatenation of the fileset linkage, the virtual device name, and the account name:

```
/fileset linked path/s<region_number>z1device1/AUTH_account id/
```

An example of the file path is:

```
/ibm/fs0/obj_sof-policy1/s13031510160z1device1/AUTH_73282e8bca894819a3cf19017848ce6b/
```

Note: To substitute the correct project prefix, see *Managing object users, roles, and projects* in *IBM Spectrum Scale: Administration Guide.*

5. List the containers defined for this account.

```
ls /ibm/fs0/obj_sof-policy1/s13031510160z1device1/AUTH_73282e8bca894819a3cf19017848ce6b/
new1    fifthcontainer  RTC73189_1  RTC73189_3  RTC73189_5  RTC73189_7  sixthcontainer
```

Administering unified file and object access

Use the following information to administer unified file and object access in your IBM Spectrum Scale setup.

Enabling the file-access object capability

Before you can use unified file and object access, you must enable the file-access object capability on the whole cluster.

• To enable the file-access object capability, enter the following command:

```
mmobj file-access enable
```

• To verify that the file-access object capability is enabled, enter the **mmobj config list** command, as in the following example:

```
mmobj config list --ccrfile spectrum-scale-object.conf --section capabilities --property file-access-enabled
```

The system displays output similar to the following:

```
file-access-enabled = true
```

Starting and stopping the ibmobjectizer service

This topic lists the commands to start and stop the ibmobjectizer service.

The ibmobjectizer service can be started and stopped by running the following commands.

Note: The ibmobjectizer service starts when **file-access-enabled** is set to true.

- To start the ibmobjectizer service when it has stopped, type `mmobj file-access enable`.
- To stop the ibmobjectizer service, type `mmobj file-access disable --objectizer`.
- To check the service status of the ibmobjectizer service, type `mmces service list -v -a | grep ibmobjectizer` .

Setting up the objectizer service interval

Take the following steps to set up the objectizer service interval.

The default interval between the completion of an objectizer cycle and the starting of the next cycle is 30 minutes. However, this needs to be planned properly based on the following:

- The frequency and the number of new file ingestions that you are expecting to be objectized.
- The number of protocol nodes you have deployed.
- How quickly you need the ingested file to be objectized.

Note: Objectization is a resource intensive process. The resource utilization is related to the number of containers that have unified file and object access enabled. The schedule of running the objectization process must be planned carefully. Running it too frequently might impact your protocol node's resource utilization. It is recommended to either schedule it during off business hours, especially if you have a small number of protocol nodes (say 2) with basic resource configuration, or schedule with an interval of 30 minutes or more if you have protocol nodes with adequate resources (where the number of protocol nodes > 2). **It is recommended to set the objectizer service interval to a minimum of 30 minutes or more irrespective of your setup.** If you need to urgently objectize files then you can use the **mmobj file-access** command that allows you to immediately objectize the specified files.

- Set up the objectization interval using the **mmobj config change** as follows.

```
mmobj config change --ccrfile spectrum-scale-objectizer.conf \
--section DEFAULT --property objectization_interval --value 2400
```

  This command sets an interval of 40 minutes between the completion of an objectization cycle and the start of the next cycle.

- Verify that the objectization time interval is changed using the **mmobj config list** as follows.

```
mmobj config list --ccrfile spectrum-scale-objectizer.conf --section DEFAULT
--property objectization_interval
```

Enabling and disabling QOS

This topic lists the commands to enable and disable QOS.

The periodic scans run by the ibmobjectizer service are resource intensive and might affect the object IO performance. Quality Of Service (QOS) can be set on the ibmobjectizer service depending upon the IO workload and the priority at which the ibmobjectizer service must be run. The usage of resources is limited to the given number so that other high priority workflows and processes can continue with adequate resources, thereby maintaining the performance of the system.

- To enable QOS, type `mmchqos <fs> --enable`.
- Set the **qos_iops_target** parameter in the spectrum-scale-objectizer.conf file. The following example is on one line:

```
mmobj config change --ccrfile spectrum-scale-objectizer.conf --section DEFAULT --property
qos_iops_target --value 400
```

- To disable QOS on ibmobjectizer, set the **qos_iops_target** to 0. The following example is on one line:

```
mmobj config change --ccrfile spectrum-scale-objectizer.conf --section DEFAULT --property
qos_iops_target --value 0
```

Configuring authentication and setting identity management modes for unified file and object access

You can configure authentication and set the identity management modes for unified file and object access using the following steps.

The identity management modes for unified file and object access are set in the `object-server-sof.conf` file. The default mode is `local_mode`.

Note: It is important to understand the identity management modes for unified file and object access and set the mode you want accordingly. Although it is possible to move from one mode to another, some considerations apply in that scenario.

The `unified_mode` identity management mode for unified file and object access is supported only with Active Directory (AD) with UNIX-mapped domains and LDAP authentication configurations. This mode must not be configured with local or user-defined authentication configurations.

Important: If you are using `unified_mode`, the authentication for both file and object access must be configured and the authentication schemes must be the same and configured with the same server. If not, the request to create object might fail with user not found error.

Use the following steps on a protocol node to configure authentication and enable `unified_mode`.

1. Determine which authentication scheme best suits your requirements. You can use either LDAP or AD with UNIX-mapped domains.

   Note: Because object can be configured with only one AD domain, you need to plan which of the UNIX-mapped AD domains, in case there are trusted domains, is to be configured for object.

2. Configure file access using the **mmuserauth** command as follows.

   ```
   mmuserauth service create --data-access-method file
   --type ad --servers myADserver --idmap-role master
   --netbios-name scale --unixmap-domains 'DOMAIN(5000-20000)'
   ```

3. Configure object access using the **mmuserauth** command as follows.

   ```
   mmuserauth service create --data-access-method object –type ad
   --user-name "cn=Administrator,cn=Users,dc=IBM,dc=local"
   --base-dn "dc=IBM,DC=local" --ks-dns-name c40bbc2xn3 --ks-admin-user admin
   --servers myADserver --user-id-attrib cn --user-name-attrib sAMAccountName
   --user-objectclass organizationalPerson --user-dn "cn=Users,dc=IBM,dc=local"
   --ks-swift-user swift
   ```

4. Change **id_mgmt** in the `object-server-sof.conf` file using the **mmobj config change** command as follows.

   ```
   mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
   --property id_mgmt --value unified_mode
   ```

5. If object authentication is configured with AD, set **ad_domain** in the `object-server-sof.conf` file.

   ```
   mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
   --property ad_domain --value POLLUX
   ```

   Note: Do not specify **ad_domain** with LDAP configurations.

   To find the correct **ad_domain** name, use the following command:

   ```
   /usr/lpp/mmfs/bin/net ads lookup -S {AD_SERVER_NAME | AD_SERVER_IP} -d0
   ```

   For example, in the output of the following command, the value of the **Pre-Win2k Domain** field is the **ad_domain**.

```
/usr/lpp/mmfs/bin/net ads lookup -S 192.196.79.34 -d0

...
Forest:   pollux.com
Domain:   pollux.com
Domain Controller: win2k8.pollux.com
Pre-Win2k Domain: POLLUX
Pre-Win2k Hostname: WIN2K8
Server Site Name :  Default-First-Site-Name
Client Site Name :  Default-First-Site-Name
...
```

Your unified file and object access enabled fileset is now configured with `unified_mode`.

6. List the currently configured **id_mgmt** mode using the **mmobj config list** command as follows.

   ```
   mmobj config list --ccrfile object-server-sof.conf --section DEFAULT --property id_mgmt
   ```

Important:

1. If the PUT requests fail in **unified_mode**, check if the user name is resolvable on the protocol nodes using the following command:

   ```
   id '<user_name>'
   ```

   If user name in AD is in the domain\user_name format, use the following command:

   ```
   id '<domain\><user_name>'
   ```

2. Ensure that the **ad_domain** parameter is not present in the `object-server-sof.conf` file when LDAP is configured.

   - To list the `object-server-sof.conf` file contents, use the following command:

     ```
     mmobj config list --ccrfile object-server-sof.conf
     ```

   - If **ad_domain** is present, remove it as follows:

     a. Copy `/etc/swift/object-server-sof.conf` to a temporary location, say /tmp.

     b. Modify the temporary file by appending a '-' before the **ad_domain** parameter. This marks that parameter for deletion.

     c. Upload the modified file using the following command:

        ```
        mmobj config change --ccrfile object-server-sof.conf --merge-file /tmp/object-server-sof.conf
        ```

     d. **[Optional]:** Validate that **ad_domain** is removed from the `object-server-sof.conf` file by listing the file contents.

3. Configuring file authentication with the same scheme as that of object authentication is a mandatory prerequisite before you enable the **unified_mode** identity management mode. In case you configure file authentication later, you must restart swift on the file server for the changes to be effective. You can do this by changing **id_mgmt** to `local_mode` and then changing it back to `unified_mode` using the following commands.

   ```
   mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
   --property id_mgmt --value local_mode
   mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
   --property id_mgmt --value unified_mode
   ```

Creating or using a unified file and object access storage policy

Use the following steps to create or use a unified file and object access storage policy.

1. Create a unified file and object access storage policy using the **mmobj policy create** command. This step also creates a fileset.

   For example:

   ```
   mmobj policy create sof-policy1 --enable-file-access
   ```

   The system displays output similar to the following:

```
[I] Getting latest configuration from ccr
[I] Creating fileset gpfs0:obj_sof-policy1
[I] Creating new unique index and building the object rings
[I] Updating the configuration
[I] Uploading the changed configuration
```

2. List the available storage policies using the **mmobj policy list** command and determine which policies are for unified file and object access by viewing the **Functions** column of the output.

   For example:

   ```
   mmobj policy list --verbose
   ```

   The system displays output similar to the following:

   ```
   Index         Name         Deprecated  Fileset        Fileset Path                        Functions               Function Details
   ---------------------------------------------------------------------------------------------------------------------------------
   0           SwiftDefault             object_fileset  /ibm/cesSharedRoot/object_fileset
   11751509160 sof-policy1              obj_sof-policy1 /ibm/cesSharedRoot/obj_sof-policy1  file-and-object-access  regions="1"
   11751509230 mysofpolicy              obj_mysofpolicy /ibm/cesSharedRoot/obj_mysofpolicy  file-and-object-access  regions="1"
   11751510260 Test19                   obj_Test19      /ibm/cesSharedRoot/obj_Test19                               regions="1"
   ```

3. Start using one of these storage policies to create data in a unified file and object access environment.

   For more information, see the following:

   - "Associating containers with unified file and object access storage policy"
   - "Creating exports on container associated with unified file and object access storage policy"

   For information on mapping of storage policy and filesets, see "Mapping of storage policies to filesets" on page 244.

You must create export at the container level. From NFS or SMB, if you create a peer container, base containers created from NFS and SMB cannot be multiprotocol.

Associating containers with unified file and object access storage policy

Use the following steps to associate a container with a unified file and object access storage policy.

1. Export common environment variables by sourcing the openrc file:

   ```
   source ~/openrc
   ```

2. Associate a container with a unified file and object access storage policy using the following command.

   ```
   swift post container1 --header "X-Storage-Policy: sof-policy1"
   ```

   In this **swift post** example, the storage policy is specified with the customized header X-Storage-Policy using the --header option.

3. Upload an object in the container associated with the unified file and object access storage policy using the following command.

   ```
   swift upload container1 imageA.JPG
   ```

Note: The steps performed using **swift** commands can also be done using **curl**. For more information, see "curl commands for unified file and object access related user tasks" on page 271.

Creating exports on container associated with unified file and object access storage policy

Use the following steps to create an NFS or SMB export on the directory that maps to the container associated with the unified file and object access storage policy.

Create an SMB or NFS export on the directory that maps to the container associated with the unified file and object access storage policy.

1. Create the NFS export as follows:

   ```
   mmnfs export add "/ibm/gpfs0/obj_sofpolicy1/s69931509221z1device1/AUTH_763476384728498323747/cont"
   ```

2. Create the SMB share as follows:

   ```
   mmsmb export add smbexport "/ibm/gpfs0/obj_sofpolicy1/s69931509221z1device1/AUTH_763476384728498323747/cont"
   ```

Note:

- It is strongly recommended that you create file exports on or below the container path level and not above it. Creating file exports above the container path level might lead to deletion of the unified file and object access enabled containers which is undesirable.
- When using POSIX interface, it is strongly recommended to only allow access of data to POSIX users from on or below the container path. Accidental deletion of container or data above might lead to inconsistent state of the system.

Enabling object access for selected files

Use the following steps to objectize files under all the containers associated with the unified file and object access storage policy under an account

In a unified file and object access environment, if you want to access files created from file interfaces such as POSIX, NFS, or CIFS through object interfaces such as curl or swift, you need to make these files available for the object interface. For making these files available for the object interface, the **ibmobjectizer** service, once activated, runs periodically and makes newly created files available for the object interface. You can also use the **mmobj file-access** command to selectively enable files for access through the object interface immediately without waiting for the objectization time interval.

The purpose of this command is to make certain files available to object sooner (or immediately) than when the objectizer would have made them available. This command does not ensure synchronization between file and object data. Therefore, files deleted are not immediately reflected in the object interface. Complete synchronization is done by the **ibmobjectizer** service eventually.

In unified file and object access enabled filesets, files can be accessed from the object interface if you know the entire URI (including keystone account ID, device etc.) to access that file without the need for them to be objectized either using the **ibmobjectizer** service or the **mmobj file-access** command.

Note: Disabling object access for files is not supported.

- To objectize files under all the containers associated with the unified file and object access storage policy under an account, use the mmobj file-access command as follows:

  ```
  mmobj file-access --storage-policy sof_policy --account-name admin
  ```

  The system displays output similar to the following:

  ```
  Loading objectization configuration from CCR
  Fetching storage policy details
  Performing objectization
  Objectization complete
  ```

  This command objectizes all containers in the account admin and enables them for access through the object interface.
- To objectize files under a container, use the mmobj file-access command as follows:

  ```
  mmobj file-access --storage-policy sof_policy --account-name admin --container-name container1
  ```

  This command objectizes all files in container1 and enables them for access through the object interface.
- To objectize a file while specifying a storage policy, use the **mmobj file-access** command with the --storage-policy option as follows:

  ```
  mmobj file-access --storage-policy sof_policy --account-name admin \
  --container-name container1 --object-name file1.txt
  ```

  This command objectizes file1.txt in container1 and enables it for access through the object interface.
- To objectize a file, use the **mmobj file-access** command as follows:

  ```
  mmobj file-access --object-path \
  /ibm/gpfs0/obj_sofpolicy1/s69931509221z1device1/AUTH_7634763384728498323747/cont/file1.txt
  ```

  This command objectizes file1.txt at location /ibm/cesSharedRoot/fileset1/Auth_12345/container1/ and enables it for access through the object interface.

For more information about the **mmobj file-access** command, see *mmobj command* in the *IBM Spectrum Scale: Command and Programming Reference.*

Example scenario - administering unified file and object access

The following example describes an end-to-end scenario of administering and using unified file and object access.

Before you can use the following steps, IBM Spectrum Scale for object storage must be installed.

This example provides a quick reference of steps performed for unified file and object access. For detailed information about these steps, see "Administering unified file and object access" on page 260.

1. Enable the file-access object capability as follows.

   ```
   mmobj file-access enable
   ```

2. [Optional] Change the objectizer service interval as follows.

   ```
   mmobj config change --ccrfile spectrum-scale-objectizer.conf \
   --section DEFAULT --property objectization_interval --value 600
   ```

3. [Optional] Change the identity management mode to `unified_mode` as follows.

   ```
   mmobj config change --ccrfile object-server-sof.conf \
   --section DEFAULT --property id_mgmt --value unified_mode
   ```

4. [Optional] Set the `ad_domain` parameter as follows.

   ```
   mmobj config change --ccrfile object-server-sof.conf \
   --section DEFAULT --property ad_domain --value ADDOMAINX
   ```

5. Create a unified file and object access storage policy as follows.

   ```
   mmobj policy create SwiftOnFileFS --enable-file-access
   ```

   The system displays output similar to the following:

   ```
   [I] Getting latest configuration from ccr
   [I] Creating fileset /dev/gpfs0:obj_SwiftOnFileFS
   [I] Creating new unique index and building the object rings
   [I] Updating the configuration
   [I] Uploading the changed configuration
   ```

   This command also creates a unified file and object access enabled fileset.

6. Create a base container with a unified file and object access storage policy as follows.

   ```
   swift post unified_access -H "X-Storage-Policy: SwiftOnFileFS"
   ```

7. Store the path created for the container by finding it in the newly created fileset as follows.

   ```
   export FILE_EXPORT_PATH=`find  /ibm/gpfs0/obj_SwiftOnFileFS/
    -name "unified_access"`

   # echo $FILE_EXPORT_PATH
   /ibm/gpfs0/obj_SwiftOnFileFS/s10041510210z1device1/
   AUTH_09271462d54b472c82adecff17217586/unified_access
   ```

8. Create an SMB share on the path as follows.

   ```
   mmsmb export add unified_access $FILE_EXPORT_PATH
   ```

   The system displays output similar to the following:

   ```
   mmsmb export add: The SMB export was created successfully
   ```

9. Create an NFS export on the path.

   ```
    mmnfs export add  $FILE_EXPORT_PATH --client \
   "*(Access_Type=RW,Squash=no_root_squash,SecType=sys)"
   ```

   The system displays output similar to the following:

   ```
   192.0.2.2:  Redirecting to /bin/systemctl stop  nfs-ganesha.service
   192.0.2.3:  Redirecting to /bin/systemctl stop  nfs-ganesha.service
   192.0.2.2:  Redirecting to /bin/systemctl start  nfs-ganesha.service
   192.0.2.3:  Redirecting to /bin/systemctl start  nfs-ganesha.service
   NFS Configuration successfully changed. NFS server restarted on all NFS nodes.
   ```

Note: If this is the first NFS export added to the configuration, the NFS service will be restarted on the CES nodes where the NFS server is running. Otherwise, no NFS restart is required when adding an NFS export.

10. Check the NFS and SMB shares.

    `mmnfs export list`

    The system displays an output similar to the following:

```
Path Delegations Clients
----------------------------------------------------------------------------
/ibm/gpfs0/obj_SwiftOnFileFS/
s10041510210z1device1/
AUTH_09271462d54b472c82adecff17217586/unified_access none *
mmsmb export list

export          path                              guest ok    smb encrypt
unified_access  /ibm/gpfs0/obj_SwiftOnFileFS/
s10041510210z1device1/
AUTH_09271462d54b472c82adecff17217586/unified_access    no          auto

Information:
The following options are not displayed because they do not contain a value:
  "browseable"
```

11. Access this export with NFS or SMB clients and create a sample directory and a file. For example: `DirCreatedFromGPFS/File1.txt` and `DirCreatedFromSMB/File2.txt`

    You can view the association of ownership when data is created from the SMB interface as follows.

```
ls -l /ibm/gpfs0/obj_SwiftOnFileFS/s10041510210z1device1/
AUTH_09271462d54b472c82adecff17217586/unified_access/DirCreatedFromSMB
total 0
-rwxr--r--. 1 ADDOMAINX\administrator
ADDOMAINX\domain users 20 Oct 21 18:09 File2.txt

mmgetacl /ibm/gpfs0/obj_SwiftOnFileFS/s10041510210z1device1/
AUTH_09271462d54b472c82adecff17217586/unified_access/DirCreatedFromSMB
#NFSv4 ACL
#owner:ADDOMAINX\administrator
#group:ADDOMAINX\domain users
special:owner@:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE
 (X)READ_ACL  (X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (X)DELETE_CHILD (X)CHOWN
 (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED

special:group@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE
 (X)READ_ACL  (X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN
 (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED

special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE
 (X)READ_ACL  (X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN
 (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

    You can view the container and the file created from the REST interface and retention of ownership in the PUT operation as follows.

```
ls -l /ibm/gpfs0/obj_SwiftOnFileFS/s10041510210z1device1/
AUTH_09271462d54b472c82adecff17217586/unified_access/DirCreatedFromSMB/File2.txt

-rwxr-xr-x. 1 ADDOMAINX\administrator ADDOMAINX\domain users 520038360 Nov 3 11:47
/ibm/gpfs0/obj_SwiftOnFileFS/s10041510210z1device1/AUTH_09271462d54b472c82adecff17217586/
DirCreatedFromSMB/unified_access/File2.txt
```

12. Objectize that file immediately by using the following command or wait for the objectization cycle to complete.

```
mmobj file-access --object-path \
/ibm/gpfs0/obj_SwiftOnFileFS/s10041510210z1device1/AUTH_09271462d54b472c82adecff17217586
/unified_access/File2.txt
```

13. List the contents of the container using the Swift client which is configured with all variables as follows.

    `swift list unified_access`

The system displays output similar to the following:

```
DirCreatedFromGPFS/File1.txt
DirCreatedFromSMB/File2.txt
```

14. Download that object using the Swift client which is configured with all variables as follows.

```
swift download unified_access/File2.txt
```

Note: The steps performed using **swift** commands can also be done using **curl**. For more information, see "curl commands for unified file and object access related user tasks" on page 271.

In-place analytics using unified file and object access

Use the following information to leverage in-place object data analytics using unified file and object access.

Unified file and object access is one of the key features of IBM Spectrum Scale for object storage that enables direct object access as files from the traditional file access such as POSIX, NFS or SMB and vice versa. Using object storage policies for containers you can have object ingested in IBM Spectrum Scale for object storage be accessed as files as well as allow files ingested using file protocols available for object access. This feature enables data analytics of object data hosted on IBM Spectrum Scale, where you can leverage in-place object data analytics. IBM Spectrum Scale supports Hadoop connectors using which you can run analytics on the object data which is accessible from the file interface and generates in-place results which are directly accessible from the object interface. This prevents any movement of data across object interfaces and thus proves to be a suitable platform for object storage as well as integrated in-place analytics for the data hosted by it.

The following diagram shows an IBM Spectrum Scale object store with unified file and object access. The object data is available as file on the same fileset. IBM Spectrum Scale Hadoop connectors allow the data to be directly leveraged for analytics.
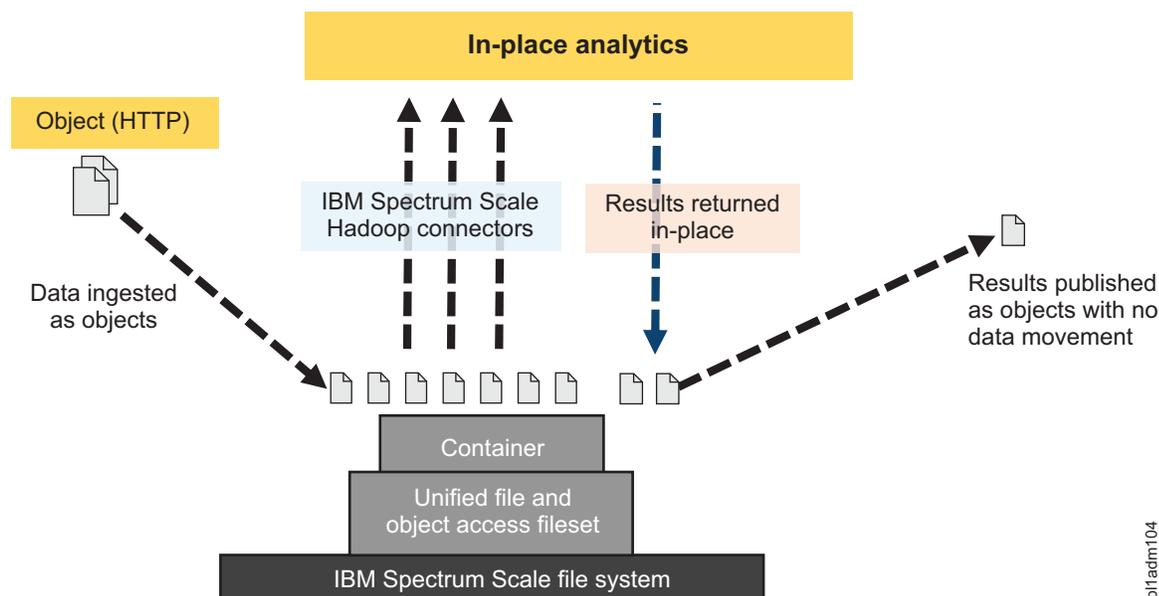

Figure 7. In-place analytics with unified file and object access

Limitations of unified file and object access

The following limitations apply for unified file and object access in IBM Spectrum Scale.

- The base container must be created from the object interface in the fileset being used for the unified access storage policy and then only the files that are added after that are enabled for object access.
- Concurrent access to the same object or file from file and object interface at the same time will lead to an undefined state. There are a variety of ways to prevent conflicts. For example:
  1. You can have your workflow enforce this.
  2. You can explicitly enforce read-only access for some periods. With NFS and SMB, it can be done in the export definition. With POSIX, it can be done using ACLs.
- Files or directories created at the base container level cannot be enabled for object access. Only the files created under the container are enabled for object access.
- Multi-region object deployment cannot be used with unified file and object access.
- Object versioning is not supported with unified file and object access.
- Special files such as device files and pipes, and soft links can exist in the object container directory, but they are not visible from the object interface.
- Containers must be deleted from the object interface. Container directories deleted from the file interface continue to show up in the container listing, until the container is deleted from the object interface.
- GPFS quota and Swift container and account quota are mutually exclusive in Release 4.2 and later. The user quota assigned to a user or a group in GPFS does not relate to the container quota defined in the object interface.
- Swift large object support (dynamic large object and static large object) is not available with unified file and object-enabled containers. S3 multipart uploads are also not supported with unified file and object-enabled containers.
- GPFS immutability is not supported with unified file and object access.
- Only object metadata can be viewed and modified from the object interface. Extended attributes defined from the file interface cannot be viewed from the object interface.
- Empty directories created from the file interface within a container are not objectized and they are not listed in the container listing.
- Files or directories with "::" or newline characters in their names are not supported and these files or data residing in these containers are not objectized.
- Change of authentication scheme of file or object could directly impact access to existing file or object data. Therefore, change of authentication is not supported as it results in loss of access for the users to the existing data on the system.
- Object ETag is inaccurate in the following scenarios:
  – Whenever an object is modified from the file interface. In this case, performing a conditional request using 'If-Match' or 'If-None-Match' headers returns incorrect results.
  – ETag for files on an explicit GET request when the size of the object has changed (increased or decreased).
  – ETag for files on an explicit GET request when the content of the object has changed but the size has remained the same.
  – If the `user.swift.metadata` extended attribute is explicitly deleted form the file interface, ETag is not present because of which the headers do not return correct results. Users need to wait for at least one cycle of objectization or they need to explicitly objectize that file to use the ETag conditional request feature.

  Note: An incorrect ETag is corrected when a GET or HEAD request is performed on the object.
- The IBM Spectrum Scale ILM policy rules work with file-extended attributes, and rules can be easily created based on extended attributes and their values. However, these rules do not work directly over Swift user-defined metadata. All of Swift user-defined metadata is stored in a single extended attribute in the IBM Spectrum Scale file system. To create ILM rules, the format and sequence in which the attributes are stored must be noted. Rules can then be created by constructing wildcard-based filters.

- To enable object access for the existing filesets, SELinux must be in the Permissive or Disabled mode.
- The conditional request, such as `If-Match` and `If-None-Match` when used with `swift` or `curl` client that performs ETag comparison does not work for the existing data enabled for object access by using the `mmobj file-access link-fileset` command . If the `--update-listing` option is used, the feature can be used after the objectizer service interval.
- If the `swift` and `curl` clients report successful container deletion after a `delete` operation is triggered on a container that has linked filesets, the directory corresponding to the container along with symlinks of the linked filesets is not deleted and must be deleted manually.

Constraints applicable to unified file and object access

The following constraints are applicable while creating and accessing objects and containers for unified file and object access:

- The name of the container must not exceed 255 characters.
- The name of the object must not exceed 214 characters.
- The path name of the object must not include successive forward slashes.
- The name of the container and the object must not be a single period (.) or a double period (..). However, a single period or a double period can be part of the name of the container and the object.

The system returns 400 Bad Request when the above constraints are not met.

The swift constraints listed in the following table are also applicable to unified file and object access.

Table 22. Configuration options for [swift-constraints] in swift.conf

Option	Limit
MAX_FILE_SIZE	5497558138880 (5 TiB)
MAX_META_NAME_LENGTH	128
MAX_META_VALUE_LENGTH	256
MAX_META_COUNT	90
MAX_META_OVERALL_SIZE	4096
MAX_HEADER_SIZE	8192
CONTAINER_LISTING_LIMIT	10000
ACCOUNT_LISTING_LIMIT	10000
MAX_ACCOUNT_NAME_LENGTH	256
VALID_API_VERSIONS	["v1", "v1.0"]
EXTRA_HEADER_COUNT	0

Note: These values can be changed by using `mmobj config change` command for the swift.conf file in swift-constraints section.

Data ingestion examples

Use the following example steps for data ingestion in the following scenarios.

- Data ingestion through object interface and access through file interface
- Data ingestion through file interface and access through object interface
- Data ingestion and access through object and file interfaces concurrently

1. **Standard REST client step:** Get proper authentication token from the Authentication URL using proper credentials to authorize on further requests.

2. **Standard REST client step:** Using token obtained in the previous step perform PUT, POST, DELETE, COPY (object only), HEAD operations for objects under container created with unified file and object access storage policy.

3. **Standard file client step:** Mount SMB or NFS exports on respective NFS or SMB clients with regular mount commands or interface available with file clients. For example:

```
mount -t cifs -o username=STORAGE5TEST\\fileuser1,password=Passw0rd5,vers=3.0 //192.0.2.4/unified_access /mnt/unified_access
```

Data ingestion using `curl`

In the following data ingestion example steps performed using **curl**, this setup is assumed:

- User: "fileuser"
- Password: "Password6"
- Account: "admin"
- Host: specscaleswift.example.com

1. Obtain the auth token using the following command:

```
curl -s -i -H "Content-Type: application/json"
-d '{ "auth": {"identity": {"methods": ["password"],"password": {"user": {"name": "fileuser","domain":
{ "name": "Default" },"password": "Passw0rd6"}}},"scope": {"project": {"name": "admin","domain": { "name": "Default" }}}}'
http://specscaleswift.example.com:35357/v3/auth/tokens
```

The auth token obtained in this step must be stored in the *$AUTH_TOKEN* variable.

2. Obtain the project list using the following command:

```
curl -s -H "X-Auth-Token:  $AUTH_TOKEN" http://specscaleswift.example.com:35357/v3/projects
```

The project ID obtained in this step must be stored in the *$AUTH_ID* variable.

3. Perform a PUT operation using the following command:

```
curl -i -s -X PUT --data @/tmp/file.txt -H "X-Auth-Token:
$AUTH_TOKEN" "http://specscaleswift.example.com:8080/v1/AUTH_$AUTH_ID/RootLevelContainer/TestObj.txt"
```

This command uploads the /tmp/file.txt file.

4. Set up the metadata age of the uploaded object using the following command:

```
curl -i -s -X POST -H "X-Auth-Token: $AUTH_TOKEN" -H
X-Container-Meta-Age:21 http://specscaleswift.example.com:8080/v1/AUTH_$AUTH_ID/RootLevelContainer/TestObj.txt
```

5. Read the metadata using the following command:

```
curl -i -s --head -H  "X-Auth-Token: $AUTH_TOKEN"
http://specscaleswift.example.com:8080/v1/AUTH_$AUTH_ID/RootLevelContainer/TestObj.txt
```

curl commands for unified file and object access related user tasks

Use the following curl commands to perform user tasks related to unified file and object access.

For the following commands, it is assumed that:

- A token is generated and it is exported as an environment variable *AUTH_TOKEN*.
- A swift endpoint URL for the project (tenant) for which token has been generated.
- A unified file and object access storage policy named `SwiftOnFileFS` is already created.

1. Create a container named `unified_access` with unified file and object access storage policy using **curl** as follows.

    ```
    curl -v -i -H "X-Auth-Token: $AUTH_TOKEN"
    -X PUT http://specscaleswift.example.com:8080/v1/AUTH_cd1a29013b6842939a959dbda95835df/unified_access/
    -H "X-Storage-Policy: SwiftOnFileFS"
    ```

    In this command, `http://specscaleswift.example.com:8080/v1/`
    `AUTH_cd1a29013b6842939a959dbda95835df/` is the endpoint URL for a project (tenant) using which a container `unified_access` is created with `SwiftOnFileFS` as the storage policy.

2. Upload an object in the container associated with the unified file and object access storage policy using **curl** as follows.

```
curl -v -i -H "X-Auth-Token: $AUTH_TOKEN"
-X PUT http://specscaleswift.example.com:8080/v1/AUTH_cd1a29013b6842939a959dbda95835df
/unified_access/object1
--data-binary @imageA.jpg
```

3. Download object residing in the unified file and object access container using **curl** as follows.

```
curl -v -i -H "X-Auth-Token: $AUTH_TOKEN"
-X GET http://specscaleswift.example.com:8080/v1/AUTH_cd1a29013b6842939a959dbda95835df
/unified_access/samplefile.txt
```

4. List the contents of the unified file and object access container using **curl** as follows.

```
curl -v -i -H "X-Auth-Token: $AUTH_TOKEN"
-X GET http://specscaleswift.example.com:8080/v1/AUTH_cd1a29013b6842939a959dbda95835df
/unified_access/
```

Configuration files for IBM Spectrum Scale for object storage

Use the following information to manage options in configuration files that are used for IBM Spectrum Scale for object storage including the unified file and object access feature. These configuration files are located in the /etc/swift directory.

For information on changing an option in a configuration file, see "Changing options in configuration files" on page 275.

object-server-sof.conf

- Contains identity management modes for unified file and object access (**id_mgmt**)
- Contains AD domain name (**ad_domain**) if AD is configured

Table 23. Configurable options for [DEFAULT] in object-server-sof.conf

Configuration option = Default value	Description
id_mgmt = local_mode	Defines the object server behavior while assigning user or group ownership to newly created objects, when those are accessed using the file interface. The allowed values are local_mode and unified_mode With local_mode, the new objects are owned by the swift user. In unified_mode, the identity of the user making the PUT request is fetched from the configured directory server.
ad_domain	When using Active Directory (AD), defines the AD domain from which the user identity should be fetched when object server is operating in the unified_mode identity management mode. **Note:** When you clean up object authentication, you must manually remove this entry. For more information, see "Configuring authentication and setting identity management modes for unified file and object access" on page 262.
tempfile_prefix = .ibmtmp_	The prefix to be used for the temporary file being created when a file being uploaded.
disable_fallocate = true	Overrides the default swift fallocate behavior, and relies on the GPFS fallocate features, excludes 'fast fail' checks.
disk_chunk_size = 65536	The size of chunks to read/write to disk (needs be equal to the file system block size).
network_chunk_size = 65536	The size of chunks to read/write over the network (needs be equal to the file system block size).
log_statsd_host = localhost	If not set, the StatsD feature is disabled.

Table 23. Configurable options for [DEFAULT] in object-server-sof.conf (continued)

Configuration option = Default value	Description
log_statsd_port = 8125	The port number for the StatsD server.
log_statsd_default_sample_rate = 1.0	Defines the probability of sending a sample for any given event or timing measurement.
log_statsd_sample_rate_factor = 1.0	Not recommended to set this to a value less than 1.0. If the frequency of logging is too high, tune the **log_statsd_default_sample_rate** instead.
log_statsd_metric_prefix =	The prefix that is added to every metric sent to the StatsD server.
retain_acl = yes	Specifies whether or not to copy the ACL from an existing object. Allowed values are yes or no.
retain_winattr = yes	Specifies whether or not to copy the Windows attributes from an existing object. Allowed values are yes or no.
retain_xattr = yes	Specifies whether or not to copy the extended attributes for the user namespace from an existing object. Allowed values are yes or no.
retain_owner = yes	Specifies whether or not to copy the UID/GID owners from an existing object. Allowed values are yes or no.

Note: Files with the .ibmtmp prefix or the one configured in the object-server-sof.conf configuration file are not objectized.

When you set the retain_* options to yes, the following attributes are retained:
- The extended attributes in the user namespace except for the **user.swift.metadata** key which contains swift metadata and it is expected to be new.
- Windows attributes

When you set the retain_* options to yes, the following attributes are not retained:
- Extended attributes in system, security, and trusted namespaces.

> **Note:** These attributes are not retained in an object's copy object operation also.

Retaining ACLs, Windows attributes, file extended attributes, and ownership, when an object is PUT over an existing object in unified file and object access enabled containers depends on your specific use case and your discretion. For example, if you are using object and file access to refer to the same data content in such a way that the object protocol might completely replace the data content in such that it might be completely new content from the file interface as well, then you might choose to not retain the existing file ACL and extended attributes. For such a use case, you might change the default values to not to retain the file ACLs, extended attributes, and ownership.

Note: If you are unsure about whether to retain these attributes or not, you might want to use the default values of retaining ACLs, Windows attributes, file extended attributes, and ownership. The default values in this case are more aligned with the expected behavior in a multiprotocol setup.

spectrum-scale-object.conf
- Contains cluster or fileset configuration information
- Unique to a site

Table 24. Configurable options for [capabilities] in spectrum-scale-object.conf

Configuration option = Default value	Description
`file-access-enabled` = false	The state for the `file-access` capability. It can be either true or false.
`multi-region-enabled` = true	The state for the `multi-region` capability. This option cannot be changed.
`s3-enabled` = true	The state for the `s3` capability. This option cannot be changed.

spectrum-scale-objectizer.conf

-

    Contains the `ibmobjectizer` service configuration information

Table 25. Configuration options for [DEFAULT] in spectrum-scale-objectizer.conf.

Configuration option = Default value	Description
`objectization_tmp_dir`	The temporary directory to be used by ibmobjectizer. This must be a path on any GPFS file system. The default value is autofilled with the path of the base file system for object.
`objectization_threads` = 24	The maximum number of threads that ibmobjectizer will spawn on a node.
`batch_size` = 100	The maximum number of files that ibmobjectizer will process in a thread.
`objectization_interval` = 1800	The time interval, in seconds, between the completion of an objectization cycle and the beginning of the next objectization cycle.
`conection_timeout` = 25	The connection time out for an account, container, or object server request from ibmobjectizer,
`response_timeout` = 25	The response time out for an account, container, object server request from ibmobjectizer
`qos_iops_target` = 0	The value assigned to ibmobjectizer to limit its resource usage. Value is given in IOPS unit. For example- 100, 400, 0.

0 means infinite. |

Table 26. Configuration options for [IBMOBJECTIZER-LOGGER] in spectrum-scale-objectizer.conf.

Configuration option = Default value	Description
`log_level` = INFO	The logging level. Allowed value is one of the following: INFO, DEBUG, WARN, ERROR

object-server.conf file

-

    Used to set swift timeout values on the lock_path calls to handle GPFS delays better

Table 27. Configuration options for object-server.conf

Configuration option = Default value	Description
`partition_lock_timeout` = 10	The time-out value while the object server tries to acquire a lock on the partition path during object create. update, and delete processes. The default value is 10.

/etc/sysconfig/memcached file

- Used to improve the performance of the internal lookups in the framework

Table 28. Configuration options for /etc/sysconfig/memcached

Configuration option = Default value	Description
`MAXCONN` = 4096	The value is set to 4096 unless the current value is higher than 4096.
`CACHESIZE` = 2048	The value is set to 2048 unless the current value is higher than 2048.

proxy-server.conf file

- Used to improve the performance of the internal lookups in the framework

Table 29. Configuration options for proxy-server.conf

Configuration option = Default value	Description
`memcache_max_connections` = 8	The default value is set to 8.
`memcache_servers`	This parameter is dynamically set to the nodes that are running the object protocol for memcache to work in a clustered environment.

Changing options in configuration files

You can use the `mmobj config change` command to change the values of the options in the configuration files. For example:

- Change the value of an option in the [DEFAULT] section of the `object-server-sof.conf` file as follows:

  ```
  mmobj config change --ccrfile object-server-sof.conf --section DEFAULT
  --property OPTIONNAME --value NEWVALUE
  ```

- Change the value of an option in the [IBMOBJECTIZER-LOGGER] section of the `spectrum-scale-objectizer.conf` file as follows:

  ```
  mmobj config change --ccrfile spectrum-scale-objectizer.conf --section IBMOBJECTIZER-LOGGER
  --property OPTIONNAME --value NEWVALUE
  ```

Note: Only some options are configurable. If an option cannot be changed, it is mentioned in the respective option description.

Attention: When a configuration file is changed using these commands, it takes several seconds for the changes to be synced across the whole cluster depending on the size of the cluster. Therefore, when executing multiple commands to change configuration files, you must plan for an adequate time interval between the execution of these commands.

Backing up and restoring object storage

Snapshots are a good way to protect data from various errors and failures. Moving them to a separate backup storage system can provide better protection against catastrophic failures of the entire storage system and might even allow the data to be stored at a lower cost. This section describes the manual steps that are needed to back up and restore the object storage and its configuration information.

In the examples, the steps to back up the Keystone configuration files and database are not given. That is the user's responsibility. You can use OpenStack backup procedures for this task. For more information on OpenStack backup procedures, see Chapter 14. Backup and Recovery.

Note:

- The same version of the IBM Spectrum Protect backup-archive client must be installed on all of the nodes that are running the `mmbackup` command.
- For more information on IBM Spectrum Protect requirements for the **mmbackup** command, see *IBM Spectrum Scale requirements* in *IBM Spectrum Scale: Administration Guide*.

Backing up the object storage

All IBM Spectrum Scale Object Nodes and IBM Spectrum Protect client nodes must be available with the object file system mounted on each node when the backup is being created. The IBM Spectrum Protect server must also be available.

Store all relevant cluster and file system configuration data in a safe location outside your GPFS cluster environment. This data is essential to restoring your object storage quickly, so you might want to store it in a site in a different geographical location for added safety.

Follow these steps to back up the object storage manually:

Note: The sample file system used throughout this procedure is called **smallfs**. Replace this value with your file system name wherever necessary.

1. Back up the cluster configuration information.

   The cluster configuration must be backed up by the administrator. The following cluster configuration information is necessary for the backup:

   - IP addresses
   - Node names
   - Roles
   - Quorum and server roles
   - Cluster-wide configuration settings from mmchconfig
   - Cluster manager node roles
   - Remote shell configuration
   - Mutual ssh and rsh authentication setup
   - Cluster UID

   Note: Complete configuration information can be found in the `mmsdrfs` file.

2. Preserve disk configuration information.

   Disk configuration must also be preserved to recover a file system. The basic disk configuration information needed for a backup intended for disaster recovery is:

   - The number of disk volumes that were previously available
   - The sizes of those volumes

   To recover from a complete file system loss, at least as much disk space as was previously available is needed for restoration. It is only possible to restore the image of a file system onto replacement disks if the disk volumes available are of similar enough sizes to the originals that all data can be restored to the new disks. The following disk configuration information is necessary for the recovery:

   - Disk device names
   - Disk device sizes
   - The number of disk volumes
   - NSD server configuration
   - Disk RAID configurations
   - Failure group designations
   - The `mmsdrfs` file contents

3. Back up the GPFS™ file system configuration information.
   In addition to the disks, the file system built on those disks has the following configuration information that can be captured using the **mmbackupconfig** command:
   - Block size
   - Replication factors
   - Number and size of disks
   - Storage pool layout
   - Filesets and junction points
   - Policy rules
   - Quota information
   - Other file system attributes

   The file system configuration information can be backed up into a single file using a command similar to the following:
   ```
   mmbackupconfig smallfs -o /tmp/smallfs.bkpcfg.out925
   ```
4. Save the following IBM Spectrum Protect configuration files for each IBM Spectrum Protect client node in the same safe location outside of your GPFS cluster.

   /etc/adsm/TSM.PWD
   > Contains the client password that is needed to access IBM Spectrum Protect. This file is present only when the IBM Spectrum Protect server setting of authentication is set to on.

   /opt/tivoli/tsm/client/ba/bin/dsm.sys and
   /opt/tivoli/tsm/client/ba/bin/dsm.opt
   > Contains the IBM Spectrum Protect client configuration files.
5. Issue the **mmcesdr primary backup** command to save the Swift configuration files for all protocol nodes.

   This command stores all of the Swift configuration data in an independent fileset created on the object storage file system. This must be done before creating the global snapshot to preserve and back up the configuration data.
6. Back up the object storage content to an IBM Spectrum Protect server by running the **mmbackup** command:
   a. Create a global snapshot by running the following command:
      mmcrsnapshot <file system device> <snapshot name>.
      For example, create a snapshot that is named `objects_globalsnap1` by running the following command:
      mmcrsnapshot smallfs objects_globalsnap1
   b. Create global and local work directories by running the following commands:
      mkdir -p /smallfs0/.es/mmbackupglobal
      mkdir -p /smallfs0/.es/mmbackuplocal
   c. Issue the following command to start the snapshot-based backup:
      mmbackup <file system device> -t incremental -N <TSM client nodes> \ -g <global work directory> \ -s <local work directory> \-S <global snapshot name> --tsm-servers <tsm server> --noquote
      The \ indicates the line wrap.
      For example:
      ```
      mmbackup smallfs -t incremental -N node1,node2 \
      -g /smallfs0/.es/mmbackupglobal \
      -s /smallfs0/.es/mmbackuplocal \
      -S objects_globalsnap1 -—tsm-servers tsm1 --noquote
      ```

      where

-N   Specifies the nodes that are involved in the backup process. These nodes must be configured for the IBM Spectrum Protect server that is being used.

-S   Specifies the global snapshot name to be used for the backup.

--tsm-servers
Specifies which IBM Spectrum Protect server is used as the backup target, as specified in the IBM Spectrum Protect client configuration dsm.sys file.

There are several other parameters available for the `mmbackup` command that influence the backup process, and the speed with which its handles the system load. For example, you can increase the number of backup threads per node by using the `-m` parameter. For the full list of parameters available, see the *mmbackup command* in the *IBM Spectrum Scale: Command and Programming Reference*.

d. Issue the following command to remove the snapshot that was created in step 6a:
   `mmdelsnapshot <file system device> <snapshot name>`
   For example:
   `mmdelsnapshot smallfs objects_globalsnap1`

Restoring the object storage

You must meet the following prerequisites before beginning the recovery procedure:

1. Restore the GPFS cluster with the same node names that were used during the backup procedure
2. Restore your OpenStack Keystone server and make sure that it is operational.
3. Install Swift software on all IBM Spectrum Scale object nodes.
4. Install the IBM Spectrum Protect backup-archive client software on the IBM Spectrum Scale object nodes that were clients previously.

   Note: All IBM Spectrum Scale object nodes and IBM Spectrum Protect client nodes must be available when the object storage configuration and contents are being restored.

After you perform the prerequisite procedures, you can begin the recovery procedure.

Note: The sample file system that is used throughout this procedure is called **smallfs**. Replace this value with your file system name wherever necessary.

1. Retrieve the base file system configuration information.

   Use the `mmrestoreconfig` command to generate a configuration file that contains the details of the former file system. For example:

   ```
   mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 -F
   smallfsQueryResultFile
   ```

2. Re-create the NSDs if they are missing.

   Using the output file that is generated in the previous step as a guide, the administrator might need to re-create NSD devices for use with the restored file system. In the output file, the NSD configuration section contains the NSD information. For example:

   ```
   ######## NSD configuration ##################
   ## Disk descriptor format for the mmcrnsd command.
   ## Please edit the disk and desired name fields to match
   ## your current hardware settings.
   ##
   ## The user then can uncomment the descriptor lines and
   ## use this file as input to the -F option.
   #
   # %nsd:
   #    device=DiskName
   #    nsd=nsd8
   ```

```
#   usage=dataAndMetadata
#   failureGroup=-1
#   pool=system
#
```

If changes are needed, edit the file in a text editor and follow the included instructions to use it as input for the **mmcrnsd** command, then issue the following command:

```
mmcrnsd -F StanzaFile
```

3. Re-create the base file system.

   The administrator must re-create the initial file system. The output query file created in step 1 can be used as a guide. The following example shows the section of this file that is needed when re-creating the file system:

```
######### File system configuration #############
## The user can use the predefined options/option values
## when recreating the file system.  The option values
## represent values from the backed up file system.
#
# mmcrfs FS_NAME NSD_DISKS -j cluster -k posix -Q yes -L 4194304 --disable-fastea
  -T /smallfs -A no --inode-limit 278016#
```

4. Restore the essential file system configuration.

   The essential file system configuration can be restored to the file system that was created in the previous step by using the **mmrestoreconfig** command. For example:

```
mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925
```

5. Issue the following command to mount the object file system on all nodes:
   mmmount <file system device> -a
   For example, mount the file system with the following command:
```
mmmount smallfs -a
```

6. Restore the configuration of the IBM Spectrum Protect client nodes by copying the saved configuration files from their saved location to each IBM Spectrum Protect client node.

   a. The IBM Spectrum Protect client config files dsm.opt and dsm.sys must be restored to /opt/tivoli/tsm/client/ba/bin/.

   b. If the IBM Spectrum Protect client password file, TSM.PWD, is saved during the backup procedure, it must be restored to /etc/adsm/.

   c. Issue the following command to verify that each IBM Spectrum Protect client node can communicate with the IBM Spectrum Protect server without prompting for a password: **dsmc q sess**

7. Restore the object storage data from the IBM Spectrum Protect server. This also restores the object storage configuration data stored within the file system.

   a. Issue the **dsmc restore** command as shown to start a no-query restore on a IBM Spectrum Protect client node.
```
dsmc restore <GPFS Object path> -subdir=yes -disablenqr=no \
-servername=<tsm server> -errorlogname=<error log path>.
```
   For example:
```
dsmc restore /smallfs/ -disablenqr=no \
-servername=tsm1 -errorlogname=/tmp/object_restore.log
```

   b. When all restore jobs are completed, check the error logs. If any errors are found, correct them so that all restore operations are complete successfully.

8. Issue the following command to restore the object storage configuration data: **mmcesdr primary restore --file-config --restore**.

9. Verify that basic Swift commands (**swift stat** and **swift list**) return without error. Also, verify that the number of containers and the number of objects within those containers are as expected.

Improving recovery time

The **dsmc restore** command starts a single restore job on a single node. This job might require a long period to restore all of your object data. To improve the restore performance, you can start separate restore jobs on different IBM Spectrum Protect client nodes.

You can create separate restore jobs by splitting a single restore task into several smaller ones. One way to do this is to specify the restore path for the object data that is deeper in the IBM Spectrum Scale object path.

For example, instead of starting the restore with the root of the IBM Spectrum Scale object path, start the object restore at the virtual devices level. If you have 40 virtual devices that are configured, you might start 40 independent restore jobs to restore the object data, and distribute the jobs to the different IBM Spectrum Protect client nodes. Additionally, you start a single restore job for all of the files under the account and container path.

With this approach, care must be taken not to overload the IBM Spectrum Protect client nodes or the IBM Spectrum Protect server. You might want to experiment to determine the best mix of jobs.

For example, if there are four IBM Spectrum Scale object nodes, each with the IBM Spectrum Protect client installed and configured, you might use the following types of commands:

1. On the first IBM Spectrum Scale object node, run a restore job for each of the first 10 virtual devices by running the following commands:

```
dsmc restore /gpfs0/objectfs/o/z1device0 -subdir=yes -disablenqr=no \
-servername=tsm1
dsmc restore /gpfs0/objectfs/o/z1device1 -subdir=yes -disablenqr=no \
-servername=tsm1
#<repeat for z1device2 - z1device9>
```

2. On the second node, run a restore job for each of the next 10 virtual devices. Continue the pattern on the remaining IBM Spectrum Scale object nodes so that all the virtual devices under the o subdirectory are restored. Also, start a single restore job for all the account and container data under the ac sub-directory by running the following command:

```
dsmc restore /gpfs0/objectfs/ac -subdir=yes -disablenqr=no -servername=tsm1
```

The most efficient restore approach depends on many factors, including the number of tape drives, IBM Spectrum Protect client configuration, and network bandwidth. You might need to experiment with your configuration to determine the most optimal restore strategy.

Configuration of object for isolated node and network groups

You can configure object for isolated node and network groups.

Object needs constant network access between all the configured CES IP addresses. The standard configuration uses all the available CES IP addresses.
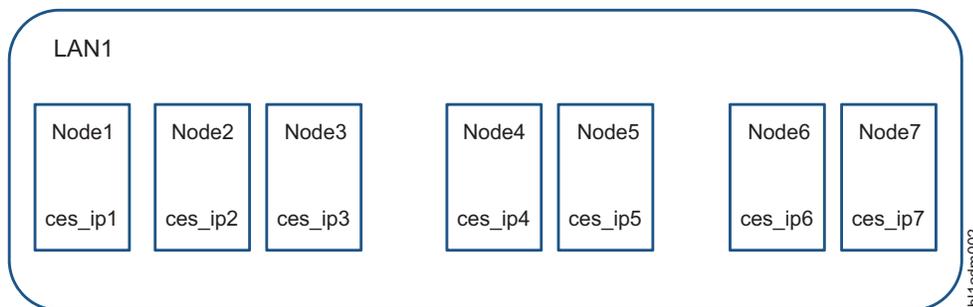
If a cluster configuration has isolated node and network groups and CES IP addresses have been assigned to those groups, parts of the object store are not accessible.
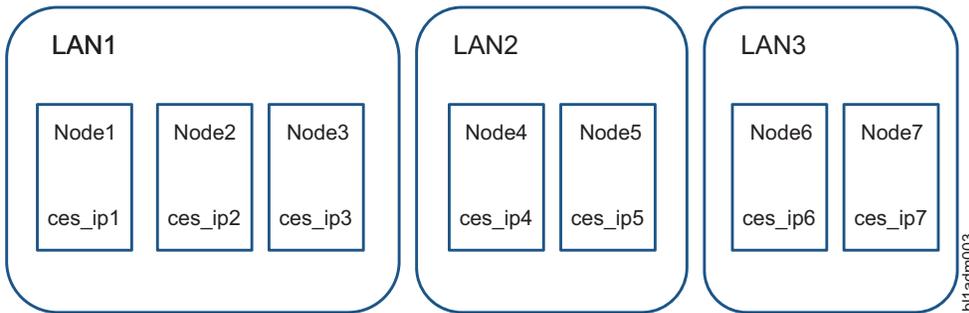
In such a configuration, a network and node group that must be used for object can be configured in the spectrum-scale-object.conf file. Only the CES IP addresses of this group are used for object.

Note: Only one object group can be used.

Note: If the Singleton and Database attributes of the IP assignments are changed manually by using the `mmces address change` command, only the IP addresses that belong to the object group can be used.

Configuration example

In the following example, LAN1 is used as the object group, IP1, IP2, and IP3 are used for object, and the object store is fully available. If only LAN1 is used, the used object services will be limited to Node1, Node2, and Node3. To distribute the object load to all the nodes in the cluster, create a network that spans across all nodes. Create an object group and assign all nodes to it. Add new CES IPs, at least one CES IP per protocol node, and assign the IPs to the same Object Group.
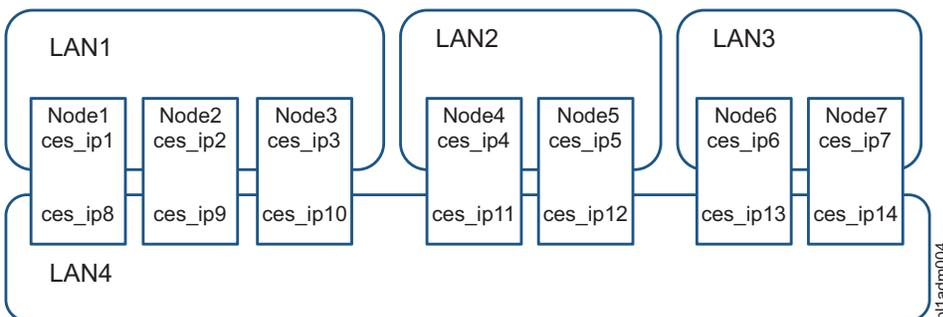
1. To set up the object group and create the LAN4 group by adding nodes to groups, run the following command:

   ```
   mmchnode --ces-group=LAN4 -N Node1,Node2,Node3,Node4,Node5,Node6,Node7
   ```

   If you want to add CES IP addresses to the group, run the following command:

   ```
   mmces address add --ces-ip ces_ip8,ces_ip9,ces_ip10,ces_ip11,
   ces_ip12,ces_ip13,ces_ip14 --ces-group LAN4
   ```

   If you want to move the existing CES IP addresses to the group, run the following command:

   ```
   mmces address change --ces-ip ces_ip8,ces_ip9,ces_ip10,ces_ip11,
   ces_ip12,ces_ip13,ces_ip14 --ces-group LAN4
   ```

2. To set up the object group when object has already been configured, run the following command. The following command is on one line:

```
mmobj config change --ccrfile spectrum-scale-object.conf --section node-group
--property object-node-group --value LAN4
```

To synchronize the ring files, run the following command:

```
/usr/lpp/mmfs/bin/mmcesobjcrring --sync
```

3. To set up the object group when object has not been configured, use the `--ces-group` option of the **mmobj swift base** command:

```
mmobj swift base  -g /gpfs/ObjectFS --cluster-hostname cluster-ces-ip.ibm --local-keystone
--enable-s3 --admin-password Passw0rd --ces-group LAN4
```

CES IPs ranging from ces_ip8 to ces_ip14 are used by the object store and the object load is distributed to all the nodes in the cluster. These IPs can be used for client connections, but more importantly, are used for traffic between the Swift proxy service and the account, container, and object services.

Enabling the object heatmap policy

Use this procedure to enable the object heatmap policy.

1. Create a file named file_heat_policy and add the following policy:

```
RULE 'DefineTiers' GROUP POOL 'TIERS'
IS 'system' LIMIT(70)
THEN 'gold' LIMIT(75)
THEN 'silver'
RULE 'Rebalance'
MIGRATE FROM POOL 'TIERS'
TO POOL 'TIERS' WEIGHT(FILE_HEAT)
FOR FILESET('Object_Fileset')
WHERE NAME LIKE '%.data'
```

This policy places the most frequently accessed objects in the SSD-backed system pool until the system pool reaches 70% of its capacity utilization. Frequently accessed objects are placed in the gold pool until it reaches 75% capacity utilization.

Note:

The temporary files generated by Swift are not moved between storage tiers because they are all eventually replaced with permanent files that have the .data extension. Moving temporary files to system, gold, or silver storage pools results in unnecessary data movement.

2. To enable the object heatmap policy for unified file and object access, identify the filename prefix for temporary files created by Swift in unified file and object access. The filename prefix is configured in object-server-sof.conf and can be fetched:

grep tempfile_prefix /etc/swift/object-server-sof.conf
```
tempfile_prefix = .ibmtmp_
```

3. Determine the filesets that are enabled for unified file and object access:

mmobj policy list
```
Index    Name          Default Deprecated Fileset      Functions
----------------------------------------------------------------------------------------
0        SwiftDefault yes            obj_fset
1317160 Sof                          obj_Sof   file-and-object-access
```

4. Create a heat based migration rule by creating the following in a file:

```
RULE 'DefineTiers' GROUP POOL 'TIERS'
      IS 'system' LIMIT(70)
      THEN 'gold' LIMIT(75)
      THEN 'silver'
RULE 'Rebalance' MIGRATE FROM POOL 'TIERS'
      TO POOL 'TIERS' WEIGHT(FILE_HEAT)
FOR FILESET('obj_Sof')
WHERE NAME NOT LIKE '.ibmtmp_%'
```

Note:

- The fileset name is derived from Step 3. Multiple fileset names can be separated by comma.
- The filename prefix in the `WHERE` clause is derived from Step 2. By using this filter, the migration of temporary files is skipped, thereby avoiding unnecessary data movement.

5. To test the policy, run the following command:

```
mmapplypolicy fs1 -P
        object_heat_policy -I test[I] GPFS Current Data Pool
        Utilization in KB and %Pool_Name KB_Occupied KB_Total
        Percent_Occupiedgold 169462784 6836715520
        2.478716330%silver 136192 13673431040
        0.000996034%system 8990720 13673431040
        0.065753211%[I] 6050 of 42706176
        inodes used: 0.014167%.[I] Loaded policy rules from
        object_heat_policy.Evaluating policy rules with
        CURRENT_TIMESTAMP = 2015-11-22@02:30:19 UTCParsed 2 policy
        rules.RULE 'DefineTiers' GROUP POOL
        'TIERS'IS 'system'
        LIMIT(70)THEN 'gold'
        LIMIT(75)THEN
    'silver'RULE 'Rebalance' MIGRATE FROM
      POOL 'TIERS' TO POOL 'TIERS'WEIGHT(computeFileHeat
              (CURRENT_TIMESTAMP-ACCESS_TIME,xattr('gpfs.FileHeat'),
              KB_ALLOCATED))FOR FILESET('Object_Fileset')
      WHERE NAME LIKE '%.data'[I] 2015-11-22@02:30:20.045
      Directory entries scanned: 1945.[I] Directories scan: 1223
      files, 594 directories, 128 other objects, 0'skipped' files and/or
      errors.[I] 2015-11-22@02:30:20.050
      Sorting 1945 file list records.[I] Inodes scan: 1223 files,
      594 directories, 128 other objects, 0 'skipped'files and/or
      errors.[I] 2015-11-22@02:30:20.345
      Policy evaluation. 1945 files scanned.[I] 2015-11-22@02:30:20.350
      Sorting 1 candidate file list records.[I] 2015-11-22@02:30:20.437
      Choosing candidate files. 1 records scanned.[I] Summary of Rule
      Applicability and File Choices:Rule# Hit_Cnt KB_Hit Chosen
      KB_ChosenKB_Ill
    Rule0 1 0 1 00 RULE 'Rebalance' MIGRATE FROM
      POOL 'TIERS' WEIGHT(.) TO POOL 'TIERS' FORFILESET(.)
      WHERE(.)[I] Filesystem objects with no
      applicable rules: 1944.[I] GPFS Policy Decisions and
      File Choice Totals:Chose to migrate 0KB: 1 of 1
      candidates;Predicted Data Pool Utilization
      in KB and %:Pool_Name KB_Occupied KB_Total
      Percent_Occupiedgold 169462784 6836715520
      2.478716330%silver 136192 13673431040
      0.000996034%system 8990720 13673431040
      0.065753211%
```

6. If there are no errors, run the following command:

```
mmapplypolicy fs1 -P object_file_heat -I yes
```

Chapter 20. Managing GPFS quotas

The GPFS quota system helps you to control the allocation of files and data blocks in a file system.

GPFS quotas can be defined for:
- Individual users
- Groups of users
- Individual filesets

Quotas are enabled by the system administrator when control over the amount of space used by the individual users, groups of users, or individual filesets is required. By default, user and group quota limits are enforced across the entire file system. Optionally, the scope of quota enforcement can be limited to an individual fileset boundaries.

Note: A large number of quota records per file system can result from the following scenarios:
- There are a very large number of users, groups, or filesets.
- If the **--perfileset-quota** option is enabled, the number of possible quota records is the number of filesets times number of users (and groups).

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Files** > **Quotas**.

Quota related tasks include:
1. "Enabling and disabling GPFS quota management"
2. "Default quotas" on page 286
3. "Explicitly establishing and changing quotas" on page 289
4. "Checking quotas" on page 292
5. "Listing quotas" on page 293
6. "Activating quota limit checking" on page 294
7. "Deactivating quota limit checking" on page 295
8. "Changing the scope of quota limit checking" on page 295
9. "Creating file system quota reports" on page 295
10. "Restoring quota files" on page 296

For GPFS fileset quotas, see "Filesets" on page 402.

Note: Windows nodes may be included in clusters that use GPFS quotas. However, Windows nodes do not support the quota commands.

Enabling and disabling GPFS quota management

You can enable GPFS quota management on new or existing GPFS file systems, establish quota values, and disable quota management by following the steps in this topic.

To enable GPFS quota management on a new GPFS file system:

1. Specify the **-Q yes** option on the **mmcrfs** command. This option automatically activates quota enforcement whenever the file system is mounted. If you want the scope of quota limit enforcement to be based on individual filesets (rather than the entire file system), also specify the **--perfileset-quota** option on the **mmcrfs** command.

2. Mount the file system.

3. Issue the **mmedquota** or **mmsetquota** command to explicitly set quota values for users, groups, or filesets. See "Explicitly establishing and changing quotas" on page 289.

To enable GPFS quota management on an existing GPFS file system:

1. Run the **mmchfs -Q yes** command. This command automatically activates quota enforcement whenever the file system is mounted or activates all subsequent mounts following the new quota setting if the file system is not mounted. If you want the scope of quota limit enforcement to be based on individual filesets (rather than the entire file system), also specify the **--perfileset-quota** option on the **mmcrfs** command.

   If an online **mmchfs -Q yes/no** command fails or is interrupted for any reason, **mmcheckquota** or **mmchfs -Q yes/no** must be rerun so that quota configuration for all nodes in the cluster will be brought into a consistent state.

   All subsequent mounts will follow the new quota setting.

   Note: The **perfileset-quota** cannot be enabled online in GPFS 4.1.

2. Compile inode and disk block statistics using the **mmcheckquota** command. See "Checking quotas" on page 292. The values obtained can be used to establish realistic quota values when issuing the **mmedquota** or **mmsetquota** command.

3. Issue the **mmedquota** or **mmsetquota** command to explicitly set quota values for users, groups, or filesets. See "Explicitly establishing and changing quotas" on page 289.

Once GPFS quota management has been enabled, you may establish quota values by:

- Setting default quotas for all new users, groups of users, or filesets.
- Explicitly establishing or changing quotas for users, groups of users, or filesets.
- Using the **gpfs_quotactl()** subroutine.

To disable quota management, run the **mmchfs -Q no** command. All subsequent mounts will obey the new quota setting.

For complete usage information, see the *mmcheckquota command*, the *mmchfs command*, the *mmcrfs command*, and the *mmedquota command* in the *IBM Spectrum Scale: Command and Programming Reference*. For additional information on quotas, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Default quotas

Default quota limits can be set for new users, groups, and filesets for a specified file system. Default quota limits can also be applied at a more granular level for new users and groups in a specified fileset.

When default quotas are managed at the fileset level, those quotas have a higher priority than those set at the file system level. If the status of the fileset-level defaults for one fileset is **Initial**, they will inherit default limits from global fileset-level defaults. The status of newly added fileset-level default quotas can be one of the following:

Initial When the fileset is created, it will be in this state. All user and group quota accounts under the fileset will not follow the fileset defaults.

Quota on
> All user and group quota accounts under the fileset that are created later will follow the fileset quota limits.

Quota off
> All user and group quota accounts under the fileset that are created later will not follow the fileset quota limits. The users and groups will follow global fileset-level defaults if they are valid. If those defaults are not valid, the status will be initial.

Specific default quota recommendations for protocols:

- Since the protocols may have vastly different fileset requirements, it is not recommended to use default quotas at the fileset level. Rather, set explicit quotas and limits for each fileset in use by any and all protocols on a case-by-case basis.
- NFS: Prepare a default quota stanza file template, and at NFS export creation time, apply the default user or group quotas to the export path (assuming the export is an independent fileset) using per-fileset default quotas.
- SMB: Prepare a default quota stanza file template, and at SMB share creation time, apply the default user or group quotas to the share path (assuming the export is an independent fileset) using per-fileset default quotas.
- Object: IBM recommends using a single independent fileset, objectfs, for the object container. See *IBM Redpaper: A Deployment Guide for IBM Spectrum Scale Object* for details. With regard to quotas, here are the relevant sections from the Redpaper™:
  - GPFS quotas: The amount of disk space and the number of inodes that are assigned as upper limits for a specified user, group of users, or fileset. With OpenStack Swift, GPFS user quotas are not used; instead, the system relies on OpenStack Swift quotas to provide a similar type of service. However, GPFS fileset quotas can still be defined (for example, for inodes, to limit the resources that are consumed by the fileset). See *Chapter 1.3 Key concepts and terminology* of the IBM Redpaper for details.
  - Swift quotas: Allows specification of the amount of disk space or number of objects that can be consumed by either an account (and subsequently all of its containers) or an individual container. The interaction between Swift quotas and GPFS quotas are described in more detail in *Chapter 6 Swift feature overview* and *Chapter 1.3 Key concepts and terminology* of the IBM Redpaper.
  - Quotas: Swift quotas allow a specific amount of disk capacity to be allocated to either containers or accounts by using Swift quotas. They also allow a limit on the maximum number of objects to be specified for containers or accounts. See *Chapter 6 Swift feature overview* of the IBM Redpaper for details.

  Note: Although GPFS quotas do not explicitly interact with Swift quotas, it still might be useful to employ GPFS quotas to limit the amount of space or the number of inodes that is consumed by the object store. To do this, define GPFS quotas on the top-level independent fileset by specifying the maximum size or maximum inode usage that the object store can consume. See *Chapter 6 Swift feature overview* of the IBM Redpaper for details.

To enable default quota values:

1. Ensure the file system is configured correctly to use quotas:
   a. The **-Q yes** option must be in effect for the file system.
   b. To set default quotas at the fileset level, the **--perfileset-quota** option must also be in effect.

   Note: If **--perfileset-quota** is in effect, all users and groups in the fileset **root** will not be impacted by default quota unless they are explicitly set.

   The **-Q yes** and **--perfileset-quota** options are specified when creating a file system with the **mmcrfs** command or changing file system attributes with the **mmchfs** command. Use the **mmlsfs** command to display the current settings of these quota options.
2. Enable default quotas with the **mmdefquotaon** command.

3. Specify default quota values for new users, groups, and filesets by issuing the **mmdefedquota** command using a default quota stanza file. A single invocation of the **mmsetquota** command using a quota stanza file can perform the following operations:

   - Set default user quotas on a file system.
   - Set default group quotas on a file system.
   - Set a default perfileset user quota on a fileset (if **--perfileset-quota** is in effect).

   The stanza file /tmp/defaultQuotaExample may look like this:

   ```
   %quota:
    device=fs1
    command=setDefaultQuota
    type=USR
    blockQuota=25G
    blockLimit=30G
    filesQuota=10K
    filesLimit=11K

   %quota:
    device=fs1
    command=setDefaultQuota
    type=GRP
    blockQuota=75G
    blockLimit=90G
    filesQuota=30K
    filesLimit=33K

   %quota:
    device=fs1
    command=setDefaultQuota
    type=USR
    fileset=fset0
    blockQuota=25G
    blockLimit=30G
    filesQuota=10K
    filesLimit=11K
   ```

   Then issue the command:

   ```
   # mmsetquota –F /tmp/defaultQuotaExample
   ```

4. To activate quota checking, use the **mmquotaon** command.
5. To list quotas, use the **mmlsquota** command:

The default quotas can be deactivated by issuing the **mmdefquotaoff** command.

For fileset recommendations, see "Filesets and quotas" on page 403.

For complete usage information, see the *mmchfs command*, *mmcrfs command*, *mmdefedquota command*, *mmdefquotaoff command*, *mmdefquotaon command*, *mmedquota command*, *mmlsfs command*, and *mmsetquota command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Implications of quotas for different protocols

Quotas can mean different things for different protocols. This section describes how quotas affect the SMB and NFS protocols.

Quotas are stored and enforced in the file system. See Chapter 20, "Managing GPFS quotas," on page 285 for details on how to enable and use quotas.

- SMB protocol and quotas

  For SMB clients, quotas can limit the used and free space reported to clients:

  – If the SMB option **gpfs:dfreequota** is set, the user quota for the current user and the group quota for the user's primary group are queried during the free space query:

- If the block limit is reached, the free space is reported as 0 and the size of the share is reported with the currently used data.
- If the soft block quota is exceeded for longer than the block grace time, the free space is reported as 0 and the size of the share is reported with the currently used data.
- If no limit is exceeded, the free space is reported as the free space according to the lowest quota limit.
- If no quota is in place, the size and free space as queried from the underlying file system are reported.
- In the case of per-fileset user and group quotas, the quotas are only queried from the root folder of the export. If a subdirectory inside the share is in a different fileset, the user and group quotas are not considered for the free space report.

Note: For including fileset quotas in the reported free space, configure the underlying file system with the --filesetdf flag (in **mmcrfs** or **mmchfs**). It is not possible to query or change individual quotas from a SMB client system.

- NFS protocol and quotas

   It is not possible to query or change individual quotas from a NFS client system. User and group quotas are not included in the reported free space to a client. To include fileset quotas in the reported space to a client, configure the underlying file system with the **--filesetdf** flag (in **mmcrfs** or **mmchfs**).

- Object protocol and quotas

   – There are two applicable levels of quotas: The quotas in the file system (see Chapter 20, "Managing GPFS quotas," on page 285) and the quotas managed by Swift.
   – Swift quotas can be used as account and container quotas. The quota values are set as account and container metadata entries. For more information, see the OpenStack Swift documentation.
   – When using file system quotas, it is important to consider that all objects stored by Swift are stored with the same owner and owning group (swift:swift).

Explicitly establishing and changing quotas

Use the **mmedquota** command to explicitly establish or change file system quota limits for users, groups of users, or filesets.

When setting quota limits for a file system, replication within the file system should be considered. See "Listing quotas" on page 293.

The **mmedquota** command opens a session using your default editor, and prompts you for soft and hard limits for blocks and inodes. For example, to set user quotas for user **jesmith**, enter:

```
mmedquota -u jesmith
```

The system displays information in your default editor similar to:

```
*** Edit quota limits for USR jesmith:
NOTE: block limits will be rounded up to the next multiple block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs0: blocks in use: 24576K, limits (soft = 0K, hard = 0K)
        inodes in use: 0, limits (soft = 0K, hard = 0K)
```

Note: A quota limit of zero indicates **no** quota limits have been established.

The current (in use) block and inode usage is for display only; it cannot be changed. When establishing a new quota, zeros appear as limits. Replace the zeros, or old values if you are changing existing limits, with values based on the user's needs and the resources available. When you close the editor, GPFS checks the values and applies them. If an invalid value is specified, GPFS generates an error message. If this occurs, reenter the **mmedquota** command. If the scope of quota limit enforcement is the entire file system, **mmedquota** will list all instances of the same user (for example, **jesmith**) on different GPFS file

systems. If the quota enforcement is on a per-fileset basis, **mmedquota** will list all instances of the same user on different filesets on different GPFS file systems.

You may find it helpful to maintain a *quota prototype*, a set of limits that you can apply by name to any user, group, or fileset without entering the individual values manually. This makes it easy to set the same limits for all. The **mmedquota** command includes the **-p** option for naming a prototypical user, group, or fileset on which limits are to be based. The **-p** flag can only be used to propagate quotas from filesets within the same file system.

For example, to set group quotas for all users in a group named **blueteam** to the prototypical values established for **prototeam**, issue:

```
mmedquota -g -p prototeam blueteam
```

You may also reestablish default quotas for a specified user, group of users, or fileset when using the **-d** option on the **mmedquota** command.

Note: You can use the **mmsetquota** command as an alternative to the **mmedquota** command.

For complete usage information, see the *mmedquota command* and the *mmsetquota command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Setting quotas for users on a per-project basis

A file system must be properly configured in order to set quotas for users. Use this information to set quotas for any number of users on a per-project basis across protocols.

1. Ensure the file system is configured correctly to use quotas:
   a. The **-Q yes** option must be in effect for the file system.
   b. To set default quotas at the fileset level, the **--perfileset-quota** option must also be in effect.

      Note: If **--perfileset-quota** is in effect, all users and groups in the fileset **root** will not be impacted by default quota unless they are explicitly set.

   The **-Q yes** and **--perfileset-quota** options are specified when creating a file system with the **mmcrfs** command or changing file system attributes with the **mmchfs** command. Use the **mmlsfs** command to display the current settings of these quota options.

   Here are some examples:
   a. A GPFS cluster is created with configuration profile file, `example.profile`, which contains the following lines:

      ```
      %filesystem
        quotasAccountingEnabled=yes
        quotasEnforced=user;group;fileset
        perfilesetQuotas=yes
      ```

      When a file system is created, those quota attributes will be set automatically. Quota accounting will be enabled on a perfileset basis for users and groups, and quotas will automatically be enforced. This means that when a quota is reached, the end user will not be able to add more data to the file system.

      mmcrfs fs5 nsd8

      A listing of the file system config, using the **mmlsfs** command, will show the following attributes and values, having been set by the **mmcrfs** command:

      mmlsfs fs5

```
...
-Q                           user;group;fileset        Quotas accounting enabled
                             user;group;fileset        Quotas enforced
                             none                      Default quotas enabled
--perfileset-quota           Yes                       Per-fileset quota enforcement
....
```

For more information on **mmcrcluster** user-defined profiles, see *mmcrcluster command* in the *IBM Spectrum Scale: Command and Programming Reference*.

b. Whether or not a GPFS cluster was created with a configuration profile file, a GPFS file system can be created with the quota attributes to be set. This can be done by calling the configuration profile file explicitly from the command line:

mmcrfs fs6 nsd9 --profile=example

For more information on user-defined profiles, see *mmcrfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

2. Create a fileset on the file system for the project using the **mmcrfileset** command.

For example:

mmcrfileset fs5 projectX --inode-space=new

Note: It is recommended to create an independent fileset for the project.

3. Link the fileset using the **mmlinkfileset** command.

The file system, fs5, must be mounted, using the **mmmount** command. For example:

mmmount fs5 -a

mmchfs fs5 --inode-limit 400000:300000

Output:

```
Set maxInodes for inode space 0 to 400000
Fileset root changed.
```

mmlinkfileset fs5 projectX -J /gpfs/fs5/projectX

4. Create export/share using the newly created fileset as the export/share path. For more information, see the *mmnfs command* and the *mmsmb command* in the *IBM Spectrum Scale: Command and Programming Reference*. For example:

mmnfs export add /gpfs/fs5/projectX

5. If needed, specify absolute fileset inode limits using the **mmchfileset** command. A fileset inode limit is analogous to saying this is how many files and directories the project is likely to produce. This is not something that can be easily recommended. Nonetheless, here is an example of how that can be set and listed for the file system and fileset:

mmchfileset fs5 projectX --inode-limit 200000

Output:

```
Set maxInodes for inode space 1 to 200000
Fileset projectX changed.
```

mmlsfileset fs5 -L

Output:

```
Filesets in file
system 'fs5':
   Name      Id   RootInode  ParentId  Created                 Inode Space  MaxInodes  AllocInodes  Comment
   root      0    3          --        Sat Mar 28 13:40:33 2015  0    400000  310656     root         fileset
   projectX  1    524291     --        Sat Mar 28 14:54:13 2015  1    200000  100032
```

6. Now that there is a fileset limit in place, which is entirely optional, to set group quota limits on the project, that is, on fileset **projectX** on file system **fs5**, use the **mmsetquota** command. For example, if the group **groupY** will access **projectX**:

mmsetquota fs5:projectX --group groupY --block 128G --files 150K

Here, the perfileset quota needs to be enabled on **fs5** as in step 1, and the group **groupY** must have a GID (group ID) on the GPFS cluster. The **block** parameter is used to specify the maximum size of the

data on the storage device and the **files** parameter is used to specify the maximum number for files (or directories) the **groupY** is able to consume or create on **projectX**, a fileset of file system **fs5** that is exported through NFS in this example.

At this point, the quota accounting needs to be refreshed on the file system using the **mmcheckquota** command, and then a reporting of the quota limits on **projectX** can take place using the **mmrepquota** command. For example:

mmcheckquota fs5

mmrepquota fs5:projectX

Output:

Name	fileset	type	KB	Block Limits quota	limit	in_doubt	grace	File Limits files	quota	limit	in_doubt	grace
root	projectX	USR	0	0	0	0	none	1	0	0	0	none
root	projectX	GRP	0	0	0	0	none	1	0	0	0	none
groupY	projectX	GRP	0	134217728	0	0	none	0	153600	0	0	none

7. If the project grows, or shrinks, and quota changes at the group level are needed, the **mmsetquota** command can again be used to change the quotas for **groupY** on **projectX**. For example, if the expected limits for **projectX doubles**:

mmsetquota fs5:projectX --group groupY --block 256G --files 300K

mmrepquota fs5:projectX

Output:

Name	fileset	type	KB	Block Limits quota	limit	in_doubt	grace	File Limits files	quota	limit	in_doubt	grace
root	projectX	USR	0	0	0	0	none	1	0	0	0	none
root	projectX	GRP	0	0	0	0	none	1	0	0	0	none
groupY	projectX	GRP	0	268435456	0	0	none	0	307200	0	0	none

8. If the project is projected to exceed the inode limits for the fileset and file system, then these can also be adjusted upwards. For more information, see the *mmchfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Checking quotas

The **mmcheckquota** command counts inode and space usage for a file system and writes the collected data into quota files.

You must use the **mmcheckquota** command if any of the following are true:

1. Quota information is lost due to node failure.

   Node failure could leave users unable to open files or deny them disk space that their quotas should allow.

2. The *in doubt* value approaches the quota limit. To see the *in doubt* value, use the **mmlsquota** or **mmrepquota** commands.

   As the sum of the *in doubt* value and the current usage may not exceed the hard limit, the actual block space and number of files available to the user, group, or fileset may be constrained by the *in doubt* value. Should the *in doubt* value approach a significant percentage of the quota, use the **mmcheckquota** command to account for the lost space and files.

Note: Running **mmcheckquota** is also recommended (in an appropriate time slot) if the following message is displayed after running **mmrepquota**, **mmlsquota**, or **mmedquota**:

```
Quota accounting information is inaccurate and quotacheck must be run.
```

When issuing the **mmcheckquota** command on a mounted file system, negative *in doubt* values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

During the normal operation of file systems with quotas enabled (not running **mmcheckquota** online), the usage data reflects the actual usage of the blocks and inodes in the sense that if you delete files you should see the usage amount decrease. The *in doubt* value does not reflect how much the user has used already, it is just the amount of quotas that the quota server has assigned to its clients. The quota server does not know whether the assigned amount has been used or not. The only situation where the *in doubt* value is important to the user is when the sum of the usage and the *in doubt* value is greater than the user's quota hard limit. In this case, the user is not allowed to allocate more blocks or inodes unless he brings the usage down.

For example, to check quotas for the file system **fs1** and report differences between calculated and recorded disk quotas, enter:

```
mmcheckquota -v fs1
```

The information displayed shows that the quota information for **USR7** was corrected. Due to a system failure, this information was lost at the server, which recorded 0 subblocks and 0 files. The current usage data counted is 96 subblocks and 3 files. This is used to update the quota:

```
fs1: quota check found the following differences:
USR7: 96 subblocks counted (was 0); 3 inodes counted (was 0)
```

Note: In cases where small files do not have an additional block allocated for them, quota usage may show less space usage than expected.

For complete usage information, see the *mmcheckquota command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Listing quotas

The **mmlsquota** command displays the file system quota limits, default quota limits, and current usage information.

If the scope of quota limit enforcement is the entire file system, **mmlsquota -u** or **mmlsquota -g** will list all instances of the same user or group on different GPFS file systems. If the quota enforcement is on a per-fileset basis, **mmlsquota -u** or **mmlsquota -g** will list all instances of the same user or group on different filesets on different GPFS file systems.

GPFS quota management takes replication into account when reporting on and determining whether quota limits have been exceeded for both block and file usage. If either data or metadata replication is enabled, the values reported by both the **mmlsquota** command and the **mmrepquota** command may exceed the corresponding values reported by commands like **ls**, **du**, and so on. The difference depends on the level of replication and on the number of replicated file system objects. For example, if data block replication is set to 2, and if all files are replicated, then the reported block usage by the **mmlsquota** and **mmrepquota** commands will be double the usage reported by the **ls** command.

When the **mmlsquota** command is issued, negative *in doubt* values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

Display the quota information for one user, group of users, or fileset with the **mmlsquota** command. If none of the options **-g**, **-u**, or **-j** are specified, the default is to display only user quotas for the user who issues the command.

To display default quota information, use the **-d** option with the **mmlsquota** command. For example, to display default quota information for users of all the file systems in the cluster, issue this command:

```
mmlsquota -d -u
```

The system displays information similar to:

```
         Default Block Limits(KB)       │  Default File Limits
Filesystem type          quota      limit │   quota     limit Remarks
fs1        USR      5242880   6291456 │      0        0

         Default Block Limits(KB)       │  Default File Limits
Filesystem type          quota      limit │   quota     limit Remarks
fs2        USR       no default limits
```

In this example, file system **fs1** shows that the default block quota for users is set at 5 GB for the soft limit and 6 GB for the hard limit. For file system **fs2**, no default quotas for users have been established.

When **mmlsquota -d** is specified in combination with the **-u**, **-g**, or **-j** options, default file system quotas are displayed. When **mmlsquota -d** is specified without any of the **-u**, **-g**, or **-j** options, default fileset-level quotas are displayed.

If you issue the **mmlsquota** command with the **-e** option, the quota system collects updated information from all nodes before returning output. If the node to which *in-doubt* space was allocated should fail before updating the quota system about its actual usage, this space might be lost. Should the amount of space in doubt approach a significant percentage of the quota, run the **mmcheckquota** command to account for the lost space.

To collect and display updated quota information about a group named **blueteam**, specify the **-g** and **-e** options:

```
mmlsquota -g blueteam -e
```

The system displays information similar to:

```
                 Block Limits                        │ File Limits
Filesystem type      KB     quota   limit  in_doubt grace │ files quota   limit  in_doubt grace
Disk quotas for group blueteam (gid 100):
fs1       GRP   45730     52000   99000      1335  none │  411   580     990        19  none
```

For complete usage information, see the *mmlsquota command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Activating quota limit checking

Quota limit checking can be activated for users, groups, filesets, or any combination of these three.

You can have quotas activated automatically whenever the file system is mounted by specifying the quota option (**-Q yes**) when creating (**mmcrfs -Q yes**) or changing (**mmchfs -Q yes**) a GPFS file system. When creating a file system, the default is to **not** have quotas activated, so you must specify this option if you want quotas activated.

The **mmquotaon** command is used to turn quota limit checking back on if it had been deactivated by issuing the **mmquotaoff** command. Specify the file system name, and whether user, group, or fileset quotas are to be activated. If you want all three fileset quotas activated (user, group, and fileset), specify only the file system name. After quotas have been turned back on, issue the **mmcheckquota** command to count inode and space usage.

For example, to activate user quotas on the file system **fs1**, enter:

```
mmquotaon -u fs1
```

To confirm the change, enter:

```
mmlsfs fs1 -Q
```

The system displays output similar to:

```
flag value         description
---- -------------- -------------------------------------------------
 -Q  user           Quotas enforced
```

For complete usage information, see the *mmquotaon command* and the *mmlsfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Deactivating quota limit checking

During normal operation, there is no need to deactivate quota enforcement. The only reason you might have to deactivate quota enforcement is when users are denied allocation that their quotas should allow, due to loss of quota information during node failure.

If this occurs, use the **mmcheckquota** command after reactivating quotas to reconcile allocation data. When quota enforcement is deactivated, disk space and file allocations are made without regard to limits.

The **mmquotaoff** command is used to deactivate quota limit checking. Specify the file system name and whether user, group, or fileset quotas, or any combination of these three, are to be deactivated. If you want all types of quotas deactivated, specify only the file system name.

For example, to deactivate only user quotas on the file system **fs1**, enter:
```
mmquotaoff -u fs1
```

To confirm the change, enter:
```
mmlsfs fs1 -Q
```

The system displays output similar to:
```
flag value         description
---- -------------- --------------------
 -Q  group;fileset  Quotas enforced
```

For complete usage information, see the *mmquotaoff command* and the *mmlsfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Changing the scope of quota limit checking

The scope of quota enforcement is established when quotas are activated. By default, user and group quota limits are enforced across the entire file system. Optionally, the scope of quota enforcement can be limited to an individual fileset boundaries.

The scope of quota enforcement can be changed using the **mmchfs** command and specifying either the **--perfileset-quota** or **--noperfileset-quota** option as needed.

After changing the scope of quota enforcement, **mmcheckquota** must be run to properly update the quota usage information.

Creating file system quota reports

You can have GPFS prepare a quota report for a file system using the **mmrepquota** command.

The quota report lists:
1. Number of files used
2. Amount of disk space used
3. Current quota limits
4. In doubt quotas (disk space allocated but currently unaccounted for)

5. Grace period allowance to exceed the soft limit
6. Whether the quotas have been explicitly set (**e**), are default values at the file system level (**d_fsys**), are default values at the fileset level (**d_fset**), or initial values (**i**)

   The entry type also indicates whether or not default quotas are enabled for the file system (**default on** or **default off**).

Specify whether you want to list only user quota information (**-u** flag), group quota information (**-g** flag), or fileset quota information (**-j** flag) in the **mmrepquota** command. The default is to summarize all three quotas. If the **-e** flag is not specified, there is the potential to display negative usage values as the quota server may process a combination of up-to-date and back-level information. See "Listing quotas" on page 293.

If the scope of quota limit enforcement is the entire file system, **mmrepquota -u** or **mmrepquota -g** will list all users or groups on different GPFS file systems. If the quota enforcement is on a per-fileset basis, **mmrepquota -u** or **mmrepquota -g** will list all instances of the same user or group on different filesets on different GPFS file systems.

To list the group quotas (**-g** option) for all file systems in the cluster (**-a** option), and print a report with header lines (**-v** option), enter:

```
mmrepquota -g -v -a
```

The system displays information similar to:

```
*** Report for GRP  quotas on fs1
            Block Limits                              |               File Limits
Name    type    KB  quota  limit  in_doubt  grace  |  files  quota  limit in_doubt    grace entryType
system  GRP  25088      0      0    209120   none  |     32      0      0     1078     none default on
usr     GRP 435256      0      0    199712   none  |     11      0      0      899     none d_fsys
```

For complete usage information, see the *mmrepquota command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Restoring quota files

The method that is used for restoring GPFS quota files depends on the version of GPFS.

The three scenarios for restoring GPFS quota files follow.

1. The file system version is lower than 4.1.0.0.

   In scenario 1, quota files can be backed up directly by copying visible quota files and then restored using the **mmcheckquota** command. The newly-specified backup quota files are transferred from normal files to quota files (metadata). Old quota files are converted from metadata to "normal" files, so these old quota files can be deleted.

2. The file system version is 4.1.0.0 or higher, but lower than 4.1.1.0.

   In scenario 2, quota files cannot be restored using the **mmcheckquota** command.

3. The file system version is 4.1.1.0 (or higher).

   In scenario 3, quota files can be restored using the **mmcheckquota** command. Use the **mmcheckquota --backup** command to back up quota files. You can restore quota files from the former backup quota files. The **mmcheckquota** command copies data from specified backup quota files to "invisible" quota files. You cannot view or delete the original quota files. You can delete specified backup quota files only.

Additional details about the three scenarios for restoring GPFS quota files follow.

In scenarios 1 and 3:

- User, group, and fileset quota files can be restored from a backup copy of the original quota file. When restoring quota files, the backup file must be in the root directory of the GPFS file system.

  In scenario 1, if a backup copy of the original quota file does not exist, an empty file will be created when the **mmcheckquota** command is issued.

  In scenario 3, the **mmcheckquota** command does nothing and prints an error.

- The user, group, or fileset files can be restored from backup copies by issuing the **mmcheckquota** command with the appropriate options.

  1. To restore the user quota file for the file system **fs1** from the backup file **userQuotaInfo**, enter:

     ```
     mmcheckquota -u userQuotaInfo fs1
     ```

     This command must be run offline (that is, no nodes are mounted).

  2. This will restore the user quota limits set for the file system, but the usage information will not be current. To bring the usage information to current values, the command must be reissued:

     ```
     mmcheckquota fs1
     ```

In scenario 1, if you want to nullify all quota configuration and then reinitialize it, follow these steps:

1. Remove the existing quota files that are corrupted.

   a. Disable quota management:

      ```
      mmchfs fs1 -Q no
      ```

   b. Remove the **user.quota**, **group.quota**, and **fileset.quota** files.

2. Enable quota management.

   a. Issue the following command:

      ```
      mmchfs fs1 -Q yes
      ```

3. Reestablish quota limits by issuing the **mmedquota** command or the **mmdefedquota** command.

4. Gather the current quota usage values by issuing the **mmcheckquota** command.

In scenario 2, quota files do not exist externally. Therefore, use **mmbackupconfig** and **mmrestoreconfig** to restore quota configurations.

For complete usage information, see the *mmcheckquota command*, the *mmdefedquota command*, and the *mmedquota command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Chapter 21. Managing GUI users

GUI users of the IBM Spectrum Scale system can monitor, configure, and manage the IBM Spectrum Scale system.

You can manage GUI users either locally within the system or in an external authentication server such as Microsoft Active Directory (AD) or Lightweight Directory Access Protocol Server (LDAP). By default, the IBM Spectrum Scale system uses an internal authentication repository for GUI users. To use an external AD or LDAP server, you need to disable the internal user repository that is used for the GUI user management and enable the LDAP/AD repository. For more information on how to disable internal repository and enable the external repository, see "Managing GUI administrators in an external authentication server" on page 301.

Managing users locally in the IBM Spectrum Scale system

You can create users who can perform different administrative tasks on the system. Each user must be part of a user group or multiple groups that are defined on the system. When you create a new user, you assign the user to one of the default user groups or to a custom user group. User groups are assigned with predefined roles that authorize the users within that group to a specific set of operations on the GUI.

Use the **Access** > **GUI Users** page to create users and add them to a user group.

Predefined roles are assigned to user groups to define the working scope within the GUI. If a user is assigned to more than one user group, the permissions are additive, not restrictive. The predefined role names cannot be changed.

The following are the default user groups:
- **Administrator**

  Manages all functions on the system except those deals with managing users, user groups, and authentication.
- **SecurityAdmin**

  Manages all functions on the system, including managing users, user groups, and user authentication.
- **SystemAdmin**

  Manages clusters, nodes, alert logs, and authentication.
- **StorageAdmin**

  Manages disks, file systems, pools, filesets, and ILM policies.
- **SnapAdmin**

  Manages snapshots for file systems and filesets.
- **DataAccess**

  Controls access to data. For example, managing access control lists.
- **Monitor**

  Monitors objects and system configuration but cannot configure, modify, or manage the system or its resources.
- **ProtocolAdmin**

  Manages object storage and data export definitions of SMB and NFS protocols.
- **UserAdmin**

Manages access for GUI users. Users who are part of this group have edit permissions only in the Access pages of the GUI.

If a GUI node fails, the application fails over to the new node. The GUI master node fails over automatically.

The IBM Spectrum Scale system is delivered with a default GUI user named *admin*. This user is also stored in the local repository. You can log in to the system by using this user name to create additional GUI users and groups in local user repository.

Use the various controls that are available under the **Password Policy** tab of the GUI Users page to enforce strong passwords for the users. You can modify or expire password of the individual users or all the users that are created in the system. If the password is set as expired, the users will be prompted to change the password in the next login.

Note: Only users with *User Administrator* role can modify the password policy of a user.

Use the various controls that are available under the Password Policy tab of the GUI Users page to enforce strong passwords for the users. You can modify or expire password of the individual users or all the users that are created in the system. If the password is set as expired, the users will be prompted to change the password in the next login.

User groups

Users who are part of Security Administrator and User Administrator user groups can create role-based user groups where any users that are added to the group adopt the role that is assigned to that group.

Roles apply to users on the system and are based on the user group to which the user belongs. A user can be part of multiple user groups so that a single user can play multiple roles in the system. You can assign the following roles to your user groups:

- **Administrator**

  Users can access all functions on the GUI except those deals with managing users and user groups.
- **Security Administrator**

  Users can access all functions on the GUI, including managing users and user groups.
- **System Administrator**

  Users manage clusters, nodes, and alert logs.
- **Storage Administrator**

  Users manage disks, file systems, pools, and filesets.
- **Snapshot Administrator**

  Users manage snapshots for file systems, filesets.
- **Monitor**

  Users can view objects and system configuration but cannot configure, modify, or manage the system or its resources.
- **Data Access**

  Users can perform the following tasks:
  - Edit owner, group, and ACL of any file or path through the **Access** > **File System ACL** > **Files and Directories** page.
  - Edit owner, group, and ACL for a non-empty directory of a file system, fileset, NFS export, or SMB share.
  - Create or delete object containers through the **Object** > **Accounts** page.
- **Protocol Administrator**

  Users manage object storage and data export definitions of SMB and NFS protocols.

- **User Administrator**

  Users manage GUI users and user groups.

Note: Default groups are not created for the user role *User Administrator* in case the user is upgrading the IBM Spectrum Scale cluster from 4.2.0.x to a later release.

Managing GUI administrators in an external authentication server

By default, the IBM Spectrum Scale uses an internal authentication repository for the GUI administrators. To use an external AD or LDAP server to manage the GUI administrators, perform the following steps:

1. Disable the internal user repository by performing the following steps:

   a. Access the server that is running the GUI and open the following file by using a text editor:
      `/opt/ibm/wlp/usr/servers/gpfsgui/server.xml`

   b. Comment out the feature `FsccUserRepo`, which is highlighted in boldface in the following example:

   ```
   <server description="GSS GUI">
   <featureManager>
   <feature>jsp-2.2</feature>
   <feature>localConnector-1.0</feature>
   <feature>jdbc-4.0</feature>
   <feature>ssl-1.0</feature>
   <feature>servlet-3.0</feature>
   <feature>appSecurity-2.0</feature>
   <!-- <feature>usr:FsccUserRepo</feature> -->
   <feature>jndi-1.0</feature>
   </featureManager>

   [...]
   ```

2. Configure *<ldapRegistry>* element in the LDAP or AD repository. Depending on the type of the external server, the configuration element can have different attribute values. Sample configuration for AD is given in the following example:

   Active Directory Server

   ```
   <ldapRegistry id="ldap"
       host="ldapserver.mycity.mycompany.com" port="389" ignoreCase="true"
       baseDN="cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
       bindDN="cn=testuser,cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
       bindPassword="testuserpwd"
       ldapType="Microsoft Active Directory"
       sslEnabled="false">
   </ldapRegistry>
   ```

   You can also specify LDAP or AD over SSL with a certificate. To do that, enable SSL, specify SSL credentials and set port to 636. The sample configuration for AD is given in the following example:

   ```
   <ldapRegistry id="ldap"
       host="ldapserver.mycity.mycompany.com" port="636" ignoreCase="true"
       baseDN="cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
       bindDN="cn=testuser,cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
       bindPassword="testuserpwd"
       ldapType="Microsoft Active Directory"
       sslEnabled="true"
       sslRef="LDAPSSLSettings">
   </ldapRegistry>
    <ssl id="LDAPSSLSettings" keyStoreRef="LDAPKeyStore" trustStoreRef="LDAPTrustStore" />
    <keyStore id="LDAPKeyStore" location="/mypath/mystore.jks"
             type="JKS" password="mystore_password" />
   <keyStore id="LDAPTrustStore" location="/mypath/mystore.jks"
             type="JKS" password="mystore_password" />
   ```

   For more information on the advanced configuration options or for enabling SSL, see Configuring LDAP user registries in Liberty.

3. Establish the LDAP group to GUI role mapping. After the GUI server was restarted, you must review the groups to roles mapping and add/remove group to role mappings as necessary.

4. View and modify the existing group to role mappings. You can view the existing groups by using the `lsusergrp` command. Adding and removing groups can be done by using **mkusergrp** and **rmusergrp** respectively.

   Note: The commands that are used to manage the GUI administrators are not available in the same path where all other IBM Spectrum Scale commands are located. The GUI user management commands are located at the following location in the system: `/usr/lpp/mmfs/gui/cli`

5. Create a group to role mapping for initial access. For initial GUI access, you need to map one existing LDAP or AD group to the SecurityAdmin GUI role. Log on to the server that is hosting the GUI and run the following command, which maps the specified LDAP group to the GUI role SecurityAdmin.

   `# /usr/lpp/mmfs/gui/cli/mkusergrp LDAPGroup --role securityadmin`

6. After the initial setup, any additional group mappings can be created through the GUI by using the **Create Group Mapping** option that is available in the **Access** > **GUI Access** page of the IBM Spectrum Scale management GUI.

   Note: The GUI Access page is available only if an external authentication server is enabled to manage the GUI user authentication. If an internal user repository is used for GUI user authentication, the GUI displays GUI Users page to create and manage GUI users and user roles.

Chapter 22. Managing GPFS access control lists

Access control protects directories and files by providing a means of specifying who is granted access. GPFS access control lists are either traditional ACLs based on the POSIX model, or NFS V4 ACLs. NFS V4 ACLs are very different than traditional ACLs, and provide much more fine control of file and directory access. A GPFS file system can also be exported using NFS.

Management of GPFS access control lists (ACLs) and NFS export includes these topics:
- "Traditional GPFS ACL administration"
- "NFS V4 ACL administration" on page 307
- "NFS and GPFS" on page 311

Traditional GPFS ACL administration

Support for NFS V4 access control lists (ACLs) has been added to traditional ACL support. NFS V4 ACLs are very different than the traditional ones.

If you are using NFS V4 ACLs, see "NFS V4 ACL administration" on page 307. Both ACL types may coexist in a single GPFS file system.

Traditional GPFS ACLs are based on the POSIX model. Traditional GPFS access control lists (ACLs) extend the base permissions, or standard file access modes, of read (r), write (w), and execute (x) beyond the three categories of file owner, file group, and other users, to allow the definition of additional users and user groups. In addition, GPFS introduces a fourth access mode, control (c), which can be used to govern who can manage the ACL itself.

In this way, a traditional ACL can be created that looks like this:

```
#owner:jesmith
#group:team_A
user::rwxc
group::rwx-
other::--x-
mask::rwxc
user:alpha:r-xc
group:audit:r-x-
group:system:rwx-
```

In this ACL:
- The first two lines are comments showing the file's owner, **jesmith**, and group name, **team_A**
- The next three lines contain the base permissions for the file. These three entries are the minimum necessary for a GPFS ACL:
  1. The permissions set for the file owner (**user**), **jesmith**
  2. The permissions set for the owner's **group**, **team_A**
  3. The permissions set for **other** groups or users outside the owner's group and not belonging to any named entry
- The next line, with an entry type of **mask**, contains the maximum permissions allowed for any entries other than the owner (the **user** entry) and those covered by **other** in the ACL.
- The last three lines contain additional entries for specific users and groups. These permissions are limited by those specified in the mask entry, but you may specify any number of additional entries up to a memory page (approximately 4 K) in size.

Traditional GPFS ACLs are fully compatible with the base operating system permission set. Any change to the base permissions, using the **chmod** command, for example, modifies the corresponding GPFS ACL as well. Similarly, any change to the GPFS ACL is reflected in the output of commands such as **ls -l**. Note that the control (c) permission is GPFS specific. There is no comparable support in the base operating system commands. As a result, the (c) permission is visible only with the GPFS ACL commands.

Each GPFS file or directory has an *access ACL* that determines its access privileges. These ACLs control who is allowed to read or write at the file or directory level, as well as who is allowed to change the ACL itself.

In addition to an *access ACL*, a directory may also have a *default ACL*. If present, the default ACL is used as a base for the access ACL of every object created in that directory. This allows a user to protect all files in a directory without explicitly setting an ACL for each one.

When a new object is created, and the parent directory has a default ACL, the entries of the default ACL are copied to the new object's access ACL. After that, the base permissions for user, mask (or group if mask is not defined), and other, are changed to their intersection with the corresponding permissions from the mode parameter in the function that creates the object.

If the new object is a directory, its default ACL is set to the default ACL of the parent directory. If the parent directory does not have a default ACL, the initial access ACL of newly created objects consists only of the three required entries (user, group, other). The values of these entries are based on the mode parameter in the function that creates the object and the umask currently in effect for the process.

Administrative tasks associated with traditional GPFS ACLs are:
1. "Setting traditional GPFS access control lists"
2. "Displaying traditional GPFS access control lists" on page 305
3. "Changing traditional GPFS access control lists" on page 306
4. "Deleting traditional GPFS access control lists" on page 306

Setting traditional GPFS access control lists

Use the following information to set GPFS ACLs:

GUI navigation

To work with this function in the GUI, log on to the IBM Spectrum Scale GUI and select **Access** > **File System ACL**.

Use the **mmputacl** command to set the access ACL of a file or subdirectory, or the default ACL of a directory. For example, to set the ACL for a file named **project2.history**, we can create a file named **project2.acl** that contains:

```
user::rwxc
group::rwx-
other::--x-
mask::rwxc
user:alpha:r-xc
group:audit:rw--
group:system:rwx-
```

In this example,
- The first three lines are the required ACL entries setting permissions for the file's owner, the owner's group, and for processes that are not covered by any other ACL entry.
- The last three lines contain named entries for specific users and groups.

- Because the ACL contains named entries for specific users and groups, the fourth line contains the required mask entry, which is applied to all named entries (entries other than the **user** and **other**).

Once you are satisfied that the correct permissions are set in the ACL file, you can apply them to the target file with the **mmputacl** command. For example, to set permissions contained in the file **project2.acl** for the file **project2.history**, enter:

```
mmputacl -i project2.acl project2.history
```

To confirm the changes, enter:

```
mmgetacl project2.history
```

The information sent to standard output is similar to:

```
#owner:guest
#group:usr
user::rwxc
group::rwx- #effective:rw--
other::--x-
mask::rw-c
user:alpha:rwxc #effective:rw-c
group:audit:rwx- #effective:rw--
group:system:-w--
```

Although you can issue the **mmputacl** command without using the **-i** option to specify an ACL input file, and make ACL entries through standard input, you will probably find the **-i** option more useful for avoiding errors when creating a new ACL.

For complete usage information, see the *mmputacl command* and the *mmgetacl command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Displaying traditional GPFS access control lists

Use the **mmgetacl** command to display the access ACL of a file or subdirectory, or the default ACL of a directory. For example, to display the ACL for the file **project2.history**, enter:

```
mmgetacl project2.history
```

The information sent to standard output is similar to:

```
#owner:guest
#group:usr
user::rwxc
group::rwx- #effective:rw--
other::--x-
mask::rw-c
user:alpha:rwxc #effective:rw-c
group:audit:rwx- #effective:rw--
group:system:-w--
```

The first two lines are comments displayed by the **mmgetacl** command, showing the owner and owning group. All entries containing permissions that are not allowed (because they are not set in the mask entry) display with a comment showing their effective permissions.

For complete usage information, see the *mmgetacl command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Applying an existing traditional GPFS access control list

To apply the same traditional ACLs from one file or directory to another:

1. Issue the **mmgetacl** command with the **-o** option to place the information in an output file.
2. Apply the ACLs to the new file or directory by issuing the **mmputacl** command with the **-i** option.

For example, use the **-o** option to specify a file to which the ACL is written:

```
mmgetacl -o old.acl project2.history
```

Then, to assign the same permissions to another file, **project.notes**, enter:

```
mmputacl -i old.acl project.notes
```

To confirm the changes, enter:

```
mmgetacl project.notes
```

The information sent to standard output is similar to:

```
#owner:guest
#group:usr
user::rwxc
group::rwx- #effective:rw--
other::--x-
mask::rw-c
user:alpha:rwxc #effective:rw-c
group:audit:rwx- #effective:rw--
group:system:-w--
```

For complete usage information, see the *mmgetacl command* and the *mmputacl command* in the *IBM Spectrum Scale: Command and Programming Reference.*

Changing traditional GPFS access control lists

Use the **mmeditacl** command to change or create the traditional ACL of a file or directory, or the default ACL of a directory. For example, to interactively edit the ACL for the file **project2.history**, enter:

```
mmeditacl project2.history
```

The current ACL entries are displayed using the default editor, provided that the EDITOR environment variable specifies a complete path name. When the file is saved, the system displays information similar to:

```
mmeditacl: 6027-967 Should the modified ACL be applied? (yes) or (no)
```

After responding **yes**, the ACLs are applied.

For complete usage information, see the *mmeditacl command* in the *IBM Spectrum Scale: Command and Programming Reference.*

Deleting traditional GPFS access control lists

Use the **mmdelacl** command to delete the extended entries in a traditional ACL of a file or directory, or the default ACL of a directory. For example, to delete the ACL for the directory **project2**, enter:

```
mmdelacl project2
```

To confirm the deletion, enter:

```
mmgetacl project2
```

The system displays information similar to:

```
#owner:uno
#group:system
user::rwxc
group::r-x-
other::--x-
```

You cannot delete the base permissions. These remain in effect after this command is executed.

For complete usage information, see the *mmdelacl command* and the *mmgetacl command* in the *IBM Spectrum Scale: Command and Programming Reference*.

NFS V4 ACL administration

AIX does not allow a file system to be NFS V4 exported unless it supports NFS V4 ACLs. By contrast, Linux does not allow a file system to be NFS V4 exported unless it supports POSIX ACLs.

This is because NFS V4 Linux servers handle NFS V4 ACLs by translating them into POSIX ACLs. For more information, see "Linux ACLs and extended attributes" on page 336.

Note:

This topic does not refer to the NFS Server function included with CES. For information, see "Authorizing protocol users" on page 316.

With AIX, the file system must be configured to support NFS V4 ACLs (with the **-k all** or **-k nfs4** option of the **mmcrfs** or **mmchfs** command). The default for the **mmcrfs** command is **-k all**.

With Linux, the file system must be configured to support POSIX ACLs (with the **-k all** or **-k posix** option of the **mmcrfs** or **mmchfs** command).

Depending on the value (**posix | nfs4 | all**) of the **-k** parameter, one or both ACL types can be allowed for a given file system. Since ACLs are assigned on a per-file basis, this means that within the same file system one file may have an NFS V4 ACL, while another has a POSIX ACL. The type of ACL can be changed by using the **mmputacl** or **mmeditacl** command to assign a new ACL or by the **mmdelacl** command (causing the permissions to revert to the mode which is in effect a POSIX ACL). At any point in time, only a single ACL can be associated with a file. Access evaluation is done as required by the ACL type associated with the file.

NFS V4 ACLs are represented in a completely different format than traditional ACLs. For detailed information on NFS V4 and its ACLs, refer to *NFS Version 4 Protocol* and other information found in the Network File System Version 4 (nfsv4) section of the IETF Datatracker website (datatracker.ietf.org/wg/nfsv4/documents).

In the case of NFS V4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual ACL entries can be flagged as being *inherited* (either by files, directories, both, or neither). Consequently, specifying the **-d** flag on the **mmputacl** command for an NFS V4 ACL is an error.

NFS V4 ACL Syntax

An NFS V4 ACL consists of a list of ACL entries. Where traditional ACLs can display one entry per line, the GPFS representation of NFS V4 ACL entries are three lines each, due to the increased number of available permissions beyond the traditional **rwxc**.

The first line has several parts separated by colons (':').
- The first part identifies the user or group.
- The second part displays a **rwxc** translation of the permissions that appear on the subsequent two lines.
- The third part is the ACL type. NFS V4 provides both an *allow* and *deny* type.

  allow   Means to allow (or permit) those permissions that have been selected with an 'X'.

  deny   Means to not allow (or deny) those permissions that have been selected with an 'X'.
- The fourth and final part is a list of flags indicating *inheritance*.

  Valid flag values are:

DirInherit

Indicates that the ACL entry should be included in the initial ACL for subdirectories created in this directory (as well as the current directory).

FileInherit

Indicates that the ACL entry should be included in the initial ACL for files created in this directory.

Inherited

Indicates that the current ACL entry was derived from inherit entries in an NFS v4 ACL of the parent directory.

InheritOnly

Indicates that the current ACL entry should *not* apply to the directory, but *should* be included in the initial ACL for objects created in this directory.

NoPropagateInherit

Indicates that the ACL entry should be included in the initial ACL for subdirectories created in this directory but not further propagated to subdirectories created below *that* level.

As in traditional ACLs, users and groups are identified by specifying the type and name. For example, **group:staff** or **user:bin**. NFS V4 provides for a set of special names that are not associated with a specific local UID or GID. These special names are identified with the keyword **special** followed by the NFS V4 name. These names are recognized by the fact that they end with the character '@'. For example, **special:owner@** refers to the owner of the file, **special:group@** the owning group, and **special:everyone@** applies to all users.

The next two lines provide a list of the available access permissions that may be *allowed* or *denied*, based on the ACL type specified on the first line. A permission is selected using an 'X'. Permissions that are not specified by the entry should be left marked with '-' (minus sign).

These are examples of NFS V4 ACLs.

1. An ACL entry that explicitly allows **READ**, **EXECUTE** and **READ_ATTR** to the **staff** group on a file is similar to this:

```
group:staff:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (X)READ_ATTR  (-)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

2. A Directory ACL is similar to this. It may include *inherit* ACL entries that do not apply to the directory itself, but instead become the initial ACL for any objects created within the directory.

```
special:group@:-----:deny:DirInherit:InheritOnly
 (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (X)READ_ATTR  (-)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

3. A complete NFS V4 ACL is similar to this:

```
#NFSv4 ACL
#owner:smithj
#group:staff
special:owner@:rwxc:allow:FileInherit
 (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (-)SYNCHRONIZE (X)READ_ACL  (X)READ_ATTR  (-)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (-)WRITE_NAMED

special:owner@:rwxc:allow:DirInherit:InheritOnly
 (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (-)SYNCHRONIZE (X)READ_ACL  (X)READ_ATTR  (-)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED

user:smithj:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (-)SYNCHRONIZE (X)READ_ACL  (X)READ_ATTR  (-)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

ACL entries DELETE and DELETE_CHILD

The ACL entries **DELETE** and **DELETE_CHILD** require special considerations. The effect of various combinations of the **DELETE** attribute for a file, and the **DELETE_CHILD** attribute for its parent directory, is given in Table 30.

In this table, the columns refer to the ACL entry for a given file, and the rows refer to the ACL entry for its parent directory. The various combinations of these attributes produce one of these results:

Permit
> Indicates that GPFS permits removal of a file with the combination of file and parent directory ACL entries specified. (Other permission checking may exist within the operating system as well.)

Deny Indicates that GPFS denies (does not permit) removal of a file with the combination of file and parent directory ACL entries specified.

Removal of a file includes renaming the file, moving the file from one directory to another even if the file name remains the same, and deleting it.

Table 30. Removal of a file with ACL entries DELETE and DELETE_CHILD

	ACL Allows DELETE	ACL Denies DELETE	DELETE not specified	UNIX mode bits only
ACL Allows DELETE_CHILD	Permit	Permit	Permit	Permit
ACL Denies DELETE_CHILD	Permit	Deny	Deny	Deny
DELETE_CHILD not specified	Permit	Deny	Deny	Deny
UNIX mode bits only - wx permissions allowed	Permit	Permit	Permit	Permit
UNIX mode bits only - no w or no x permissions allowed	Permit	Deny	Deny	Deny

The UNIX mode bits are used in cases where the ACL is not an NFS V4 ACL.

NFS V4 ACL translation

NFS V4 access requires that an NFS V4 ACL be returned to clients whenever the ACL is read. This means that if a traditional GPFS ACL is associated with the file, a translation to NFS V4 ACL format must be performed when the ACL is read by an NFS V4 client. Since this translation has to be done, an option (**-k nfs4**) is provided on the **mmgetacl** and **mmeditacl** commands, so that this translation can be seen locally as well.

It can also be the case that NFS V4 ACLs have been set for some file system objects (directories and individual files) prior to administrator action to revert back to a POSIX-only configuration. Since the NFS V4 access evaluation will no longer be performed, it is desirable for the **mmgetacl** command to return an ACL representative of the evaluation that will now occur (translating NFS V4 ACLs into traditional POSIX style). The **-k posix** option returns the result of this translation.

Users may need to see ACLs in their true form as well as how they are translated for access evaluations. There are four cases:

1. By default, the **mmgetacl** command returns the ACL in a format consistent with the file system setting:
   - If **posix** only, it is shown as a traditional ACL.
   - If **nfs4** only, it is shown as an NFS V4 ACL.
   - If **all** formats are supported, the ACL is returned in its true form.
2. The command **mmgetacl -k nfs4** always produces an NFS V4 ACL.
3. The command **mmgetacl -k posix** always produces a traditional ACL.

4. The command **mmgetacl -k native** always shows the ACL in its true form, regardless of the file system setting.

In general, users should continue to use the **mmgetacl** and **mmeditacl** commands without the **-k** flag, allowing the ACL to be presented in a form appropriate for the file system setting. Since the NFS V4 ACLs are more complicated and therefore harder to construct initially, users that want to assign an NFS V4 ACL should use the command **mmeditacl -k nfs4** to start with a translation of the current ACL, and then make any necessary modifications to the NFS V4 ACL that is returned.

Setting NFS V4 access control lists

There is no option on the **mmputacl** command to identify the type (traditional or NFS V4) of ACL that is to be assigned to a file. Instead, the ACL is assumed to be in the traditional format unless the first line of the ACL is:

```
#NFSv4 ACL
```

The lines that follow the first one are then processed according to the rules of the expected ACL type.

An NFS V4 ACL is similar to this:

```
#NFSv4 ACL
#owner:root
#group:system
special:owner@:rwxc:allow
(X)READ/LIST (X)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL  (-)READ_ATTR  (-)READ_NAMED
(X)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (-)WRITE_NAMED

special:owner@:-----:deny
(-)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (-)READ_ATTR  (X)READ_NAMED
(-)DELETE    (X)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (X)WRITE_NAMED

user:guest:r-xc:allow
(X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL  (-)READ_ATTR  (-)READ_NAMED
(X)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED

user:guest:-----:deny
(-)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (-)READ_ATTR  (X)READ_NAMED
(-)DELETE    (X)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED
```

This ACL shows four ACL entries (an allow and deny entry for each of **owner@** and **guest**).

In general, constructing NFS V4 ACLs is more complicated than traditional ACLs. Users new to NFS V4 ACLs may find it useful to start with a traditional ACL and allow either **mmgetacl** or **mmeditacl** to provide the NFS V4 translation, using the **-k nfs4** flag as a starting point when creating an ACL for a new file.

Displaying NFS V4 access control lists

The **mmgetacl** command displays an existing ACL regardless of its type (traditional or NFS V4). The format of the ACL that is returned depends on the file system setting (**-k** flag), as well as the format of the actual ACL associated with the file. For details, see "NFS V4 ACL translation" on page 309.

Applying an existing NFS V4 access control list

This function is identical, whether using traditional or NFS V4 ACLs. See "Applying an existing traditional GPFS access control list" on page 305.

Changing NFS V4 access control lists

This function is identical, whether using traditional or NFS V4 ACLs. See "Changing traditional GPFS access control lists" on page 306.

Deleting NFS V4 access control lists

Use the **mmdelacl** command to delete NFS V4 ACLs. Once the ACL has been deleted, permissions revert to the mode bits. If the **mmgetacl** command is then used to display the ACL (**mmgetacl -k native**), it appears as a traditional GPFS ACL.

When assigning an ACL to a file that already has an NFS V4 ACL, there are some NFS rules that must be followed. Specifically, in the case of a directory, there will **not** be two separate (access and default) ACLs, as there are with traditional ACLs. NFS V4 requires a single ACL entity and allows individual ACL entries to be flagged if they are to be inherited. Consequently, **mmputacl -d** is not allowed if the existing ACL was the NFS V4 type, since this attempts to change **only** the default ACL. Likewise **mmputacl** (without the **-d** flag) is not allowed because it attempts to change only the access ACL, leaving the default unchanged. To change such an ACL, use the **mmeditacl** command to change the entire ACL as a unit. Alternatively, use the **mmdelacl** command to remove an NFS V4 ACL, followed by the **mmputacl** command.

Considerations when using GPFS with NFS V4 ACLs

There are several constraints that you need to consider when using GPFS with NFS V4 ACLs. For a complete description of these restrictions, see "Exceptions and limitations to NFS V4 ACLs support" on page 336.

NFS and GPFS

GPFS file systems may be exported using the Network File System (NFS) protocol from one or more nodes. After export, normal access to the file system can proceed from GPFS cluster nodes or NFS client nodes.

Note: GPFS on Windows does not provide NFS integration.

See the "Authorizing protocol users" on page 316 topic for information on ACLs for protocol users.

Considerations for the interoperability of a GPFS file system include:
- "Exporting a GPFS file system using NFS"
- "NFS usage of GPFS cache" on page 315
- "Synchronous writing using NFS" on page 315
- "Unmounting a file system after NFS export" on page 315
- "NFS automount considerations" on page 315
- "Clustered NFS and GPFS on Linux" on page 316

Note: None of these sections take into account the NFS server integration that is introduced with CES. The integrated NFS server interactions, with the following documented sections, will be addressed in a future release.

Exporting a GPFS file system using NFS

To export a GPFS file system:
1. Create and mount the GPFS file system. In the examples, we assume a file system with a local mount point of **/gpfs**.

    For performance reasons, some NFS implementations cache file information on the client. Some of the information (for example, file state information such as file size and timestamp) is not kept up-to-date in this cache. The client may view stale inode data (on **ls -l**, for example) if exporting a GPFS file system with NFS.

If this is not acceptable for a given installation, caching can be turned off by mounting the file system on the client using the appropriate operating system mount option (for example, **-o noac** on Linux NFS clients). Turning off NFS caching results in extra file systems operations to GPFS, and negatively affect its performance.

Note:

- Ensure that all GPFS file systems that you use to export data via NFS are mounted with the **syncnfs** option to prevent clients from running into data integrity issues during failover. Prior to mounting a GPFS system, it is a good practice to run the **mmchfs** command to set the **syncnfs** option, **-o syncnfs**.
- Ensure that NFS clients mount with the **-o hard** option to prevent any application failures during network failures or node failovers.
- If caching is turned on on the NFS clients, files that are migrated to the cloud storage tier by using Transparent cloud tiering remain in the co-resident status, and the capacity is not freed from the file system. However, if caching is disabled, the files are moved to the non-resident status and the capacity is freed. In this case, there is a negative impact on the performance. Therefore, there is a tradeoff between capacity and performance, and administrators must take a judicious decision depending on the business requirements.

2. Make sure that the clocks of all nodes in the GPFS cluster are synchronized. If this is not done, NFS access to the data, as well as other GPFS file system operations, may be disrupted.

   NFS relies on metadata timestamps to validate the local operating system cache. If the same directory is either NFS-exported from more than one node, or is accessed with both the NFS and GPFS mount point, it is critical that clocks on all nodes that access the file system (GPFS nodes and NFS clients) are constantly synchronized using appropriate software (for example, NTP). Failure to do so may result in stale information seen on the NFS clients.

3. Ensure that NFS is properly configured and running.

   For Linux nodes, information on configuring NFS can be obtained at the linuxdocs.org websitewww.linuxdocs.org.

   For AIX nodes, refer to AIX in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/ ssw_aix/welcome) for information about configuring NFS.

4. Use the **mmnfs export add**, **mmnfs export change**, or **mmnfs export remove** command as required.

   Note: For IBM Spectrum Scale exports, you must use **mmnfs export add**, **mmnfs export change**, or **mmnfs export remove** command. Editing the **/etc/exports** file to specify file systems to be exported does not work for IBM Spectrum Scale exports; this can only be used for exports that are not IBM Spectrum Scale exports.

Export considerations

Keep these points in mind when exporting a GPFS file system to NFS. The operating system being used and the version of NFS might require special handling or consideration.

Linux export considerations: Linux does not allow a file system to be NFS V4 exported unless it supports POSIX ACLs. For more information, see "Linux ACLs and extended attributes" on page 336.

For Linux nodes only, issue the **exportfs -ra** command to initiate a reread of the **/etc/exports** file.

Starting with Linux kernel version 2.6, an **fsid** value must be specified for each GPFS file system that is exported on NFS. For example, the format of the entry in **/etc/exports** for the GPFS directory **/gpfs/dir1** might look like this:

```
/gpfs/dir1 cluster1(rw,fsid=745)
```

The administrator must assign **fsid** values subject to the following conditions:

1. The values must be unique for each file system.

2. The values must not change after reboots. The file system should be unexported before any change is made to an already assigned **fsid**.

3. Entries in the **/etc/exports** file are not necessarily file system roots. You can export multiple directories within a file system. In the case of different directories of the same file system, the **fsid**s must be different. For example, in the GPFS file system **/gpfs**, if two directories are exported (**dir1** and **dir2**), the entries might look like this:

```
/gpfs/dir1 cluster1(rw,fsid=745)
/gpfs/dir2 cluster1(rw,fsid=746)
```

4. If a GPFS file system is exported from multiple nodes, the **fsid**s should be the same on all nodes.

Configuring the directories for export with NFSv4 differs slightly from the previous NFS versions. To configure the directories, do the following:

1. Define the root of the overall exported file system (also referred to as the pseudo root file system) and the pseudo file system tree. For example, to define **/export** as the pseudo root and export **/gpfs/dir1** and **/gpfs/dir2** which are not below **/export**, run:

```
mkdir –m 777 /export /export/dir1 /export/dir2
mount --bind /gpfs/dir1 /export/dir1
mount –-bind /gpfs/dir2 /export/dir2
```

In this example, **/gpfs/dir1** and **/gpfs/dir2** are bound to a new name under the pseudo root using the bind option of the **mount** command. These bind mount points should be explicitly unmounted after GPFS is stopped and bind-mounted again after GPFS is started. To unmount, use the **umount** command. In the preceding example, run:

```
umount /export/dir1; umount /export/dir2
```

2. Edit the **/etc/exports** file. There must be one line for the pseudo root with **fsid=0**. For the preceding example:

```
/export cluster1(rw,fsid=0)
/export/dir1 cluster1(rw,fsid=745)
/export/dir2 cluster1(rw,fsid=746)
```

The two exported directories (with their newly bound paths) are entered into the **/etc/exports** file.

Large installations with hundreds of compute nodes and a few login nodes or NFS-exporting nodes require tuning of the GPFS parameters **maxFilesToCache** and **maxStatCache** with the **mmchconfig** command.

This tuning is required for the GPFS token manager (file locking), which can handle approximately 1,000,000 files in memory. The token manager keeps track of a total number of tokens, which equals **5000 * number of nodes**. This will exceed the memory limit of the token manager on large configurations. By default, each node holds 5000 tokens.

The valid range for **maxStatCache** is 0 - 100,000,000. If you do not specify values for **maxFilesToCache** and **maxStatCache**, the default value of **maxFilesToCache** is 4000 and the default value of **maxStatCache** is 1000. If you specify a value for **maxFilesToCache** but not for **maxStatCache**, the default value of **maxStatCache** is 4 * **maxFilesToCache**. On upgrades to IBM Spectrum Scale v4.1 from v3.4 or earlier, the existing defaults (1000 for **maxFilesToCache** and 4000 for **maxStatCache**) remain in effect.

Note: The stat cache is not effective on the Linux platform unless the Local Read-Only Cache (LROC) is configured. Therefore on Linux, the setting for **maxStatCache** depends on whether the Local Read-Only Cache (LROC) is enabled on the node:

- If LROC is not enabled on the node, set **maxStatCache** to 0.
- If LROC is enabled on the node, set **maxStatCache** to an appropriate nonzero value based on the number of nodes in the cluster, the number of token managers in the cluster, the size of the LROC device, and any other relevant factors. For more information about LROC, see the topic *Local read-only cache* in the *IBM Spectrum Scale: Administration Guide*.

This restriction applies to all the versions and distributions of Linux that IBM Spectrum Scale supports.

If you are running at SLES 9 SP 1, the kernel defines the **sysctl** variable **fs.nfs.use_underlying_lock_ops**, which determines whether the NFS **lockd** is to consult the file system when granting advisory byte-range locks. For distributed file systems like GPFS, this must be set to **true** (the default is **false**).

You can query the current setting by issuing the command:
```
sysctl fs.nfs.use_underlying_lock_ops
```

Alternatively, the **fs.nfs.use_underlying_lock_ops = 1** record can be added to **/etc/sysctl.conf**. This record must be applied after initially booting the node, and after each reboot, by issuing the command:
```
sysctl -p
```

Because the **fs.nfs.use_underlying_lock_ops** variable is currently not available in SLES 9 SP 2 or later, when NFS-exporting a GPFS file system, ensure that your NFS server nodes are at the SP 1 level (unless this variable is made available in later service packs).

For additional considerations when NFS exporting your GPFS file system, refer to *File system creation considerations* topic in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

AIX export considerations: AIX does not allow a file system to be exported by NFS V4 unless it supports NFS V4 ACLs.

NFS export considerations for versions prior to NFS V4: For NFS exported file systems, the version of NFS you are running with may have an impact on the number of inodes you need to cache, as set by both the **maxStatCache** and **maxFilesToCache** parameters on the **mmchconfig** command. The implementation of the **ls** command differs from NFS V2 to NFS V3. The performance of the **ls** command in NFS V3 in part depends on the caching ability of the underlying file system. Setting the cache large enough will prevent rereading inodes to complete an **ls** command, but will put more of a CPU load on the token manager.

Also, the clocks of all nodes in your GPFS cluster must be synchronized. If this is not done, NFS access to the data, as well as other GPFS file system operations, may be disrupted.

NFS V4 export considerations: For information on NFS V4, refer to *NFS Version 4 Protocol* and other information found in the Network File System Version 4 (nfsv4) section of the IETF Datatracker website (datatracker.ietf.org/wg/nfsv4/documents).

To export a GPFS file system using NFS V4, there are two file system settings that must be in effect. These attributes can be queried using the **mmlsfs** command, and set using the **mmcrfs** and **mmchfs** commands.
1. The **-D nfs4** flag is required. Conventional NFS access would not be blocked by concurrent file system reads or writes (this is the POSIX semantic). NFS V4 however, not only allows for its requests to block if conflicting activity is happening, it insists on it. Since this is an NFS V4 specific requirement, it must be set before exporting a file system.
   ```
   flag value          description
   ---- -------------- ----------------------------------------------------
    -D nfs4            File locking semantics in effect
   ```
2. The **-k nfs4** or **-k all** flag is required. Initially, a file system has the **-k posix** setting, and only traditional GPFS ACLs are allowed. To export a file system using NFS V4, NFS V4 ACLs must be enabled. Since NFS V4 ACLs are vastly different and affect several characteristics of the file system objects (directories and individual files), they must be explicitly enabled. This is done either exclusively, by specifying **-k nfs4**, or by allowing **all** ACL types to be stored.
   ```
   flag value          description
   ---- -------------- ----------------------------------------------------
    -k  all            ACL semantics in effect
   ```

Note: In IBM Spectrum Scale 4.2 and later, NFS connections are limited to a maximum of 2250 for a large number of NFS exports. The maximum number of NFS exports supported is 1000.

NFS usage of GPFS cache

Exporting a GPFS file system from a node may result in significant additional demands on the resources at that node. Depending on the number of NFS clients, their demands, and specific mount options, you may want to increase either one or both of the **maxFilesToCache** and **pagepool**. The CES NFS number of open files is set to 1 Mg. By default, gpfs `maxFilesToCache` is set to 4K. Increasing is set to 4K. If the NFS clients overrun the 4K cache by generating more files, NFS RPC's will fail due to TCP inactivity timeout. Increasing `maxFilesToCache` solves this problem.

See the **mmchconfig** command.

You may also choose to restrict the use of the NFS server node through the normal GPFS path and not use it as either a file system manager node or an NSD server.

Synchronous writing using NFS

With Linux, **write** operations are usually asynchronous. If synchronous writes are required over NFS, edit the **/etc/exports** file to include **sync,no_wdelay**.

Unmounting a file system after NFS export

Because NFS use of a GPFS file system may result in a file being held, attempting to unmount a GPFS file system may return a 'Device is busy' error. If this occurs, stop the NFS daemons before attempting to unmount the file system at the NFS server. On Linux, issue this command:

`/etc/rc.d/init.d/nfs stop`

On AIX, issue this command:

`stopsrc -g nfs`

NFS can be restarted after the unmount completes. On Linux, issue this command:

`/etc/rc.d/init.d/nfs start`

On AIX, issue this command:

`startsrc -g nfs`

NFS automount considerations

The default file system type when using the automounter daemon is NFS. When the **-fstype** option is not specified, and the server is the local node, a soft-mount of the local directory is done at the desired mount point. JFS is assumed as the only handler of local directories. A GPFS file system local soft-mount does not work implicitly, since the mount request is passed to JFS which then produces an error. When specifying **-fstype mmfs** the local soft-mount works because the mount is then passed to GPFS instead of JFS.

A GPFS soft-mount does not automatically unmount. Setting **-fstype nfs3** causes the local server mounts to always go through NFS. This allows you to have the same **auto.map** file on all nodes whether the server is local or not, and the automatic unmount will occur. If you want local soft-mounts of GPFS file systems while other nodes perform NFS mounts, you should have different **auto.map** files on the different classes of nodes. This should improve performance on the GPFS nodes as they will not have to go through NFS.

Clustered NFS and GPFS on Linux

In addition to the traditional exporting of GPFS file systems using NFS, GPFS allows you to configure a subset of the nodes in the cluster to provide a highly available solution for exporting GPFS file systems via NFS.

The participating nodes are designated as Cluster NFS (CNFS) member nodes and the entire setup is frequently referred to as CNFS or CNFS cluster.

In this solution, all CNFS nodes export the same file systems to the NFS clients. When one of the CNFS nodes fails, the NFS serving load moves from the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover.

Authorizing protocol users

Authorization grants or denies access to resources such as directories, files, commands, and functions. Authorization is applicable to an already authenticated identity, such as an IBM Spectrum Scale data user, an administrative user, or an IBM service representative. Access to the files and directories of the IBM Spectrum Scale system is managed through access control lists (ACLs). It ensures that only authorized users get access to exports, directories, and files. An access control entry (ACE) is an individual entry in an access control list, and describes the permissions for an individual user or group of users. An ACL can have zero or more ACEs.

Authorizing file protocol users

The IBM Spectrum Scale system uses ACLs to authorize users who access the system through file protocols such as NFS and SMB.

The GPFS file system supports storing POSIX and NFSv4 ACLs to authorize file protocol users.

SMB service maps the NFSv4 ACL to a Security Descriptor for SMB clients to form the ACLs. That is, the SMB ACL is derived from the NFSv4 ACL; it is not a separate ACL. Any changes from SMB clients on ACLs are mapped back to the ACLs in the file system.

To get the expected behavior of ACLs, you must configure the file system to use only the NFSv4 ACLs. The default configuration profiles (`/usr/lpp/mmfs/profiles`) that are included with IBM Spectrum Scale contain the required configuration for NFSv4 ACLs in the file system. When you manually create a file system for protocol usage with the `mmcrfs` command, use the `-k nfs4` option to establish the correct ACL setting. For more information, see the *mmcrfs command* and the *mmchfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

ACLs can be applied at the following levels:
- Files and directories
- SMB shares

ACLs on files and directories

With the SMB and NFS protocols, you can manage the ACL permissions on files and directories from connected file systems. ACLs from both protocols are mapped to the same ACL in the file system. The ACL supports inheritance and you can control the inheritance by using the special inheritance flags.

When you create the directory for an SMB or NFS export, you must ensure that the ACL on the exported folder is set up as you want. One option is to change the owner of the export folder to an administrator, who can then access the new export and change the ACL from an SMB or NFS client. This approach is a good practice to follow for setting up SMB shares for use with Windows clients.

It is a good practice to manage ACLs from the ACL management interface on a connected client system. For example, after you create the directories for an SMB or NFS export, you can set the owner of the directory with the **chown** command. Then the user can connect to the export with the SMB or NFS protocol to see and manage the ACLs that are associated with the directory over which the export is created.

ACLs and POSIX mode bits

The POSIX bits of a file or directory are another authorization method, different from ACLs. POSIX bits can also be used to specify access permissions for a file. You can use the POSIX bits of a file to configure access control for an owner, a group, and for all users to read, update, or run the file. POSIX bits are less flexible than ACLs.

Changing the POSIX mode bits also modifies the ACL of an object. When you use ACLs for access control, it is a good practice to ensure that ACLs are not replaced with permissions from POSIX mode bits. You can configure this behavior with the **--allow-permission-change** parameter in the **mmcrfileset** and **mmchfileset** commands.

An ACL extends the base permissions or the standard file access modes such as read, write, and execute. ACLs are compatible with UNIX mode bits. Issuing the **chmod** command by the NFS clients overwrite the access privileges that are defined in the ACL by the privileges that are derived from UNIX mode bits. By default, the ACLs are replaced by UNIX mode bits if the **chmod** command is submitted. To allow proper use of ACLs, it is a good practice to prevent **chmod** from overwriting the ACLs by setting this parameter to **setAclOnly** or **chmodAndSetAcl**.

NFSv3 clients can set and read the POSIX mode bits; NFSv3 clients who set the UNIX permissions modify the ACL to match the UNIX permissions. In most NFS-only cases, the POSIX permissions are used directly. For NFSv3 clients, file sharing with SMB access protection is done by using NFSv4 ACLs but NFSv3 clients can see only the mapping of ACLs to traditional UNIX access permissions. The full NFSv4 ACLs are enforced on the server.

The permissions from the NFSv4 ACL entries special:owner@ are shown as the POSIX permission bits for the file owner, special:group@ are shown as the POSIX permission bits for the group, and special:everyone@ are shown for the POSIX permissions for "other".

SMB protocol export-level ACLs

SMB share ACLs apply only to SMB exports and they are separate from the file system ACLs. When you export ACLs, users need to have access in the share-level ACL and in the file or directory.

SMB share ACLs can be changed either through the MMC on a Windows client or through the **mmsmb exportacl** command. For more information, see the **mmsmb exportacl** command on any protocol node.

Mapping between SMB protocol Security Descriptors and NFSv4 ACLs stored in the file system

The SMB protocol uses Security Descriptors to describe the permissions on a file or directory. The Discretionary Access Control Lists (DACL) from the Security Descriptors are mapped to and from the NFSv4 ACLs stored in the file system. That way, only one authoritative ACL exists for each file or directory.

Note: With automatic ID mapping (autorid), Samba assigns a UID and a GID with the same numeric value to each SID. This causes SMB Security Descriptor to be always mapped to NFSv4 ACL group entry as SIDs do not distinguish between a user and a group.

The mapping of entries from an SMB Security Descriptor to a NFSv4 ACL is done according to the following table:

Table 31. Mapping from SMB Security Descriptor to NFSv4 ACL entry

Entry in SMB Security Descriptor		Mapped to NFSv4 ACL entry	
Principal	Inheritance	Principal	Inheritance
EVERYONE	Any	special:everyone@	Same
CREATOR OWNER	Subfolders or files	special:owner@	Same with InheritOnly added
CREATOR GROUP	Subfolders or files	special:group@	Same with InheritOnly added
(Principal matching owner of file or directory)	None	special:owner@	None
	Subfolders or files	Explicit entry for owning user, not special:owner@	Same
(Principal matching owning group of file or directory)	None	special:group@	None
		Explicit entry for owning group, not special:group@	Same
(Other principal)	Any	Explicit entry	Same

The mapping from the NFSv4 ACL to an SMB Security Descriptor is done according to the following table:

Table 32. Mapping from NFSv4 ACL entry to SMB Security Descriptor

Entry in NFSv4 ACL		Mapped to SMB Security Descriptor entry		
Principal	Inheritance	Principal	Inheritance	Comment
special:everyone@	Any	EVERYONE	Same	(No comment)
special:owner@	Includes FileInherit or DirInherit	CREATOR OWNER	Same with the exclusion of the current file or folder added	An entry that applies to the current file or folder and marked for inheritance can result in mapping one NFS4 entry in two Security Descriptor entries.
	Applies to current folder (not InheritOnly)	Mapped user	Same	
special:group@	Includes FileInherit or DirInherit	CREATOR GROUP	Same with the exclusion of the current file or folder added	An entry that applies to the current file or folder and marked for inheritance can result in mapping one NFS4 entry in two Security Descriptor entries.
	Applies to current folder (not InheritOnly)	Mapped group	Same	
Explicit entry	Any	Matching principal	Same	(No comment)

Independent of this mapping, Microsoft Windows Clients expect the ACL to be stored in a canonical order. To avoid problems with presenting differently ordered ACLs to these clients, manage the ACLs for SMB clients that run Microsoft Windows from a Microsoft Windows system.

ACL inheritance

The inheritance flags in ACL entry of parent directories are used to control the inheritance of authorization to the child files and directories. The inheritance flag gives you the granularity to specify whether the inheritance defined in an ACL entry applies to the current directory and its children or only to the subdirectories and files that are contained in the parent directory. ACL entries are inherited to the child directories or files at the time of creation. Changes made to the ACL of a parent directory are not propagated to child directories or files. However, in case of SMB, you can specify to propagate the inheritance changes from a parent to all its child by using File Explorer, command line, or PowerShell.

Controlling inheritance of entries inside an ACL

The NFSV4 protocol uses the following flags to specify and control inheritance information of the ACEs:

- *FileInherit:* Indicates that this ACE must be added to each new non-directory file created. This flag is signified by 'f' or file_inherit.
- *DirInherit:* Indicates that this ACE must be added to each new directory created. This flag is signified by 'd' or `dir_inherit`.
- *InheritOnly:* Indicates that this ACE is not applied to the parent directory itself, but only inherited by its children. This flag is signified by 'i' or inherit_only.
- *NoPropagateInherit:* Indicates that the ACL entry must be included in the initial ACL for subdirectories that are created in this directory but not further propagated to subdirectories created below that level.

In case of SMB, the following list shows how the inheritance flags are linked to the Microsoft Windows inheritance modes:

- This folder only (No bits)
- This folder, subfolder, and files (FileInherit, DirInherit)
- This folder and subfolders (DirInherit)
- This folder and files (FileInherit)
- Subfolders and files only (FileInherit, DirInherit, InheritOnly)
- Subfolders only (DirInherit, InheritOnly)
- Files only (FileInherit, InheritOnly)

ACL best practices

It is essential to properly apply ACLs to the file systems, filesets, and exports, and directories and files to ensure smooth access for the users.

Consider the following points before you create or copy data into the export:

1. Should a group of users be given permission to access the data?
2. Should individual users be given permission to access the data?
3. Should selected users from different groups be given access to selected data?
4. Should the shares be in mixed mode? That is, do you have NFS and SMB clients who access the exports in explicit mode, where the data is accessed either from SMB or NFS?
5. Should the applications that the clients are using over SMB and NFS be given any specific ACL permission?

Setting ACLs for groups

The recommended way to manage access is per group instead of per individual user. This way, users can be easily added to or removed from the group. Providing ACLs to groups has an added advantage of managing inheritance easily for the whole group of users simultaneously. If individual users are added directly to ACLs and you need to make a change, you need to update ACLs of all corresponding directories and files. On the authentication server like Active Directory or LDAP, you can create groups and add users as members and use these groups to give respective access to data.

Setting ACLs for individual users

If you need to set ACLs for individual users where data is created by users in folders that are created by others, it is recommended that you explicitly add the users who need ACLs on that export.

In mixed mode, where the share is used for both NFS and SMB access, parent owner might experience loss of access to the child directory or the files. To avoid such a problem, it is recommended that you provide ACLs explicitly to each user.

Special Owner and Group

The special owner and group dynamically refer to the owner and group of the directory or file that the ACL belongs to. For example, if the owner of a file is changed, all special:owner@ entries in the ACL refers to the new owner. In case of inheritance, this leads to some complexity because those special entries point to the owner and group of the child directory or file that inherits the entry. In many cases, the children do not have the same owner and group as the parent directory. Therefore, the special entries in parent and children refer to different users. This can be avoided by adding static entries (user:'name' or group:'name') to the ACL. These static entries are inherited by name and refer everywhere to the same users. But they are not updated if the owner of the parent is changed. The general recommendation is not to use special:owner@ and special:group@ together with inheritance flags. For more information, see the `mmputacl` command.

Setting ACLs for special IDs

The inheritance of ACL from the owner of a directory to subdirectories and files works only for subdirectories and files that have the same owner as the parent directory. A subdirectory or file that is created by a different owner does not inherit the ACL of a parent directory that is owned by another user.

In case of special access to NFSV4 exports, parent owners might experience loss of access to its child folders and files. To avoid such a problem, for mixed mode, it is recommended that you provide ACLs to groups rather than to individual users.

ACL permissions that are required to work on files and directories

The topic describes the required ACL permissions to access files and folders through file protocols.

The following table describes the ACL permissions that are required when the user of the file is not the file owner, where "X" denotes permission that is required on file or directory and "P" denotes permission that is required on the parent directory of the file or directory.

Table 33. ACL permissions required to work on files and directories, while using SMB protocol (table 1 of 2)

ACL Operation	ACL Permission					
	Traverse folder / execute file	List folder / read data	Read attribute	Read extended attribute	Create files / write data	Create folders / append data
Execute file	X	X				
List folder		X				
Read data from file		X	X	X		
Read attributes			X			
Create file					X	
Create folder						X
Write data to file		X	X		X	X
Write file attributes						

Table 33. ACL permissions required to work on files and directories, while using SMB protocol (table 1 of 2) (continued)

ACL Operation	ACL Permission					
	Traverse folder / execute file	List folder / read data	Read attribute	Read extended attribute	Create files / write data	Create folders / append data
Write folder attributes						
Delete file		P	X		P	
Delete folder		P	X		P	
Rename file		P	X		P	
Rename folder		P	X		P	P
Read file permissions						
Read folder permissions						
Write file permissions						
Write folder permissions						
Take file ownership						
Take folder ownership						

Table 34. ACL permissions required to work on files and directories, while using SMB protocol (table 2 of 2)

ACL Operation	ACL Permission						
	Write attribute	Write extended attributes	Delete subfolder and files	Delete	Read permissions	Write permissions	Take ownership
Execute file							
List folder							
Read data from file							
Read attributes							
Create file							
Create folder							
Write data to file	X	X					
Write file attributes	X						
Write folder attributes	X						
Delete file				P or X			
Delete folder				P or X			
Rename file				P or X			
Rename folder				P or X			
Read file permissions					X		
Read folder permissions					X		

Table 34. ACL permissions required to work on files and directories, while using SMB protocol (table 2 of 2) (continued)

ACL Operation	ACL Permission						
	Write attribute	Write extended attributes	Delete subfolder and files	Delete	Read permissions	Write permissions	Take ownership
Write file permissions					X	X	
Write folder permissions					X	X	
Take file ownership							X
Take folder ownership							X

Table 35. ACL permissions required to work on files and directories, while using NFS protocol (table 1 of 2)

ACL Operation	ACL Permission					
	Traverse folder / execute file	List folder / read data	Read attribute	Read extended attribute	Create files / write data	Create folders / append data
Execute file	P, X	X				
List folder	P	X				
Read data from file	P	X				
Read attributes	P					
Create file	P				P	
Create folder	P					P
Write data to file	P				X	X
Write file attributes	P					
Write folder attributes	P					
Delete file	P				P	
Delete folder	P				P	
Rename file	P		X		P	
Rename folder	P		X		P	P
Read file ACL	P					
Read folder ACL	P					
Write file ACL	P					
Write folder ACL	P					
Take file ownership	P					
Take folder ownership	P					

Table 36. ACL permissions required to work on files and directories, while using NFS protocol (table 2 of 2)

ACL Operation	ACL Permission						
	Write attribute	Write extended attributes	Delete subfolder and files	Delete	Read ACL	Write ACL	Take ownership
Execute file							
List folder							
Read data from file							
Read attributes							
Create file							
Create folder							
Write data to file							
Write file attributes							
Write folder attributes							
Delete file			P				
Delete folder			P				
Rename file			P				
Rename folder			P				
Read file ACL							
Read folder ACL							
Write file ACL						X	
Write folder ACL						X	
Take file ownership							X
Take folder ownership							X

The following are the considerations on the ACL read and write permissions:

1. The files that require "Traverse folder / execute file" permission do not require the "Bypass Traverse Check" attribute to be enabled. This attribute is enabled by default on the files.

2. The "Read extended attribute" permission is required by the SMB clients with recent Microsoft Windows versions (for Microsoft Windows 2008, Microsoft Windows 2012, and Microsoft Windows 8 versions) for file copy operations. The default ACLs set without inheritance do not contain this permission. It is recommended that you use inherited permissions where possible and enable this permission in the inherited permissions to prevent the default value to be used and cause problems.

Migrating data through SMB to the IBM Spectrum Scale cluster requires a user ID with the enhanced permissions. The ownership of a file cannot be migrated by a normal IBM Spectrum Scale user. Therefore, you need to configure an "admin user" to allow data migration. For more information on how to configure the "admin users" parameter, see the *mmsmb export add* and *mmsmb export change* sections in *mmsmb command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Directory traversal permissions that are applicable for SMB ACLs

The following are the considerations on the traverse permissions:

1. It is recommended that you add the "Traverse folder / execute file" permission to all executable files, even if the "Bypass Traverse Check" attribute is enabled on these files. IBM Spectrum Scale checks for the "Traverse folder / execute file" permission on executable files irrespective of the value of the "Bypass Traverse Check" attribute.

2. If the `--cifsBypassTraversalChecking` option is enabled, it allows a user to directly access files and folders that the user owns, and also that are contained under the parent folders for which the user does not have Read or Write permissions. Users without "Read and Execute" access to the share or export in which the user-owned files and folders are located can read and modify the files inside the export for which the user has permissions that are granted by the `--cifsBypassTraversalChecking` option. However, in this case, operations like rename file and delete file are not granted by default. This is normal SMB behavior. Modify ACLs as required to enable these operations.

   For example, in the directory structure /A/B/C, assume that an SMB user has 'read' permission on C but no permissions on A and B. When the `--cifsBypassTraversalChecking` option is set to its default value Yes, this SMB user can access C without having "Traverse Folder" or "Execute File" permissions that are set to allow on A and B, but is still not allowed to browse the content of A and B.

3. The ownership of a file cannot be migrated by a normal user. You must configure and use administrative user credentials to perform data migration. When migrating existing files and directories from other systems to IBM Spectrum Scale, the ACL might not contain explicit traversal rights for the users because the source system can grant this right implicitly. After migrating the files with ACLs, ensure that traversal rights are granted to the parent directory of each exported path.

Working with ACLs

The IBM Spectrum Scale system applies default ACLs for newly created IBM Spectrum Scale file system components such as file system, filesets, file, directories, and exports.

The file system must be created with native ACL type as NFS V4. It is recommended to use the default configuration profiles (`/usr/lpp/mmfs/profiles`) that are included with IBM Spectrum Scale. It contains the required configuration for NFSV4 ACLs in the file system.

Applying default ACLs

Perform the following steps to apply default ACLs on SMB and NFS exports:

1. Create a fileset or directory in the file system as shown in the following example:

   ```
   mkdir -p /ibm/gpfs0/testsmbexport
   ```

2. Change the owner and group of the fileset or directory using **chown** and **chgrp** respectively. For example:

   ```
   chown -R "DOMAIN\\username":"DOMAIN\\groupname" /ibm/gpfs0/testsmbexport
   ```

3. Use the **mmputacl** or **mmeditacl** commands to set the wanted ACE along with specific ACE for owner user and owner group and inheritance flags for the fileset or directory.

4. Check the ACL setting for the fileset or directory by using the **mmgetacl** command.

5. Create the desired SMB or NFS export by using the **mmnfs** or **mmsmb** commands over the fileset or directory.

6. For data exported for SMB clients, it is recommend to manage the ACLs from a Windows clients, since there is already a GUI interface available and the ACL is set according to the requirements of Windows clients. Modifying the ACL directly with **mmputacl** and **mmeditacl** are not advised.

Viewing the owner of the SMB share

Perform the following steps to create an SMB share and view the owner of the export:

1. Submit the **mmsmb export add** command to create an SMB share as shown in the following example:

   ```
   mmsmb export add testsmbexport /ibm/gpfs0/testsmbexport
   ```

2. Issue either the **ls -la** command or the **mmgetacl** command to view the owner of the export. For example:

```
ls -la /ibm/gpfs0/testsmbexport
```

Or

```
mmgetacl /ibm/gpfs0/testsmbexport
```

Apart from the tasks that are listed earlier in this section, the following table provides a quick overview of the tasks that can be performed to manage ACLs and the corresponding IBM Spectrum Scale command.

Table 37. Commands and reference to manage ACL tasks.

Tasks that can be performed to manage ACLs	Command	Reference topic
Applying ACL at file system, fileset, and export level	`mmeditacl`	"Applying an existing NFS V4 access control list" on page 310
Inserting ACEs in existing ACLs	`mmeditacl`	"Changing NFS V4 access control lists" on page 310
Modifying ACLs	`mmeditacl`	"Changing NFS V4 access control lists" on page 310
Copying Access control list entries	`mmeditacl`	"Changing NFS V4 access control lists" on page 310
Replacing a complete ACL	`mmputacl` or `mmeditacl`	"Changing NFS V4 access control lists" on page 310
Replacing all entries for a specific user inside an ACL	`mmeditacl`	"Changing NFS V4 access control lists" on page 310
Controlling inheritance of entries inside an ACL	`mmputacl` or `mmeditacl`	
Deleting complete ACL	`mmdelacl`	"Deleting NFS V4 access control lists" on page 311
Deleting specific ACL entries	`mmeditacl`	"Changing NFS V4 access control lists" on page 310
Deleting ACL entry for a user	`mmeditacl`	"Changing NFS V4 access control lists" on page 310
Displaying an ACL	`mmgetacl`	"Displaying NFS V4 access control lists" on page 310
Changing file system directory's owner and group	`chown` or `chgroup`	
Displaying file system directory's owner and group	`ls –l` or `mmgetacl`	

Important: The `mmgetacl`, `mmputacl`, and `mmeditacl` commands are available to change the ACLs directly. As the SMB clients might depend on the order of entries in the ACL, it is not recommended to change the ACLs directly on GPFS while using the SMB protocol. Changing an ACL directly in GPFS also does not account for inherited entries. So, it is recommended to change the ACLs from a windows client.

Managing ACLs from Windows clients

For SMB shares, it is recommended to manage the ACLs from a Windows client. The following operations are included in creating an SMB share:

1. Create the folder to export in the file system with the mkdir command.
2. Change the owner of the exported folder to a user who configures the initial ACLs.
3. Create the export using the `mmsmb export add` command.

4. Using a Windows client machine, access the newly created share as the user specified in step 2.
5. Right-click on the shared folder, and select **Properties**.
6. Select the **Security** tab and then select **Advanced** to navigate to the more detailed view of permissions.
7. Add and remove permissions as required.

Authorizing object users

The Object Storage service of the IBM Spectrum Scale system uses Keystone service for Identity Management. The Identity Management service consists of user authentication and authorization processes.

Access for the object users to the Object Storage projects are controlled by the user roles and container ACLs. Based on the roles defined for the user, object users can be administrative users and non-administrative users. Non-admin users can only perform operations per container based on the container's X-Container-Read and X-Container-Write ACLs. Container ACLs can be defined to limit access to objects in swift containers. Read access can be limited to only allow download, or allow download and listing. Write access allows the user to upload new objects to a container.

You can use an external AD or LDAP server or a local database as the back-end to store and manage user credentials for user authentication. The authorization details such as relation of users with projects and roles are maintained locally by the keystone server. The customer can select the authentication server to be used. For example, if AD is existing in an enterprise deployment and the users in AD are required to access object data, the customer can decide to use AD as the back-end authentication server.

When the back-end authentication server is AD or LDAP, the user management operations such as creating user, deleting user, and so on are the responsibility of the AD or LDAP administrator, who can optionally also be the Keystone server administrator. When local authentication is used for object access, the user management operations are done by the Keystone administrator. In case of authorization, the management tasks such as creating roles, projects, and associating the user with them is done by the Keystone Administrator. The Keystone administration can be done through the Keystone V3 REST API or by using an OpenStack python-based client.

Before you start creating object users, and projects, ensure that Keystone server is configured and the authentication servers are set up properly. You can use the `mmces service list -a -v` command to see whether Keystone is configured properly.

The object users are authorized to the object data and resources by creating and managing roles and ACLs. The roles and ACLs define the actions that can be performed by the user on the object resources such as accessing data, managing the projects, creating projects, read, write, run permissions, and so on.

Configuring container ACLs to authorize object data users

The following examples and sections provide an understanding on how to set up container ACLs and define the access permissions for the user.

Creating containers:

The Object Storage organizes data in account, container, and object. Each account and container is an individual database that is distributed across the cluster. An account database contains the list of containers in that account. A container database contains the list of objects in that container.

It is the responsibility of the Keystone server administrator to create and manage accounts. The account defines a namespace for containers. A container must be unique within the owning account and account must use a unique name within the project. The `admin` account is created by default.

The following is an example of how to create containers.

GUI navigation

To work with this function in the IBM Spectrum Scale GUI, log on to the GUI and select **Object** >
Containers.

1. Issue the **swift post container** command to create a container by using the Swift Command Line
   Client. In the following example, the Keystone administrator creates a public_readOnly container in
   admin account.

   ```
   # swift post public_readOnly  --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
   --os-project-name admin --os-project-domain-name Default --os-username admin
   --os-user-domain-name Default --os-password Passw0rd --auth-version 3
   ```

2. Issue the **swift list** command to list the containers that are available for the account. In the
   following example, the system lists the containers that are available in the admin project.

   ```
   # swift list --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
   --os-project-name admin --os-project-domain-name Default --os-username admin
   --os-user-domain-name Default --os-password Passw0rd --auth-version 3
   public_readOnly
   ```

3. Issue the **swift stat** command to list the accounts, containers, or objects details. In the following
   example, the system displays the admin account details.

   ```
   # swift stat -v --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
   --os-project-name admin --os-project-domain-name Default --os-username admin
   --os-user-domain-name Default --os-password Passw0rd --auth-version 3
                        StorageURL: http://tully-ces-ip.adcons.spectrum:8080/v1
   /AUTH_bea5a0c632e54eaf85e9150a16c443ce
                        Auth Token: 1f6260c4f8994581a465b8225075c932
                           Account: AUTH_bea5a0c632e54eaf85e9150a16c443ce
                        Containers: 1
                           Objects: 0
                             Bytes: 0
   Containers in policy "policy-0": 1
   Objects in policy "policy-0": 0
       Bytes in policy "policy-0": 0
      X-Account-Project-Domain-Id: default
                       X-Timestamp: 1432766053.43581
                        X-Trans-Id: tx9b96c4a8622c40b3ac69a-0055677ce7
                      Content-Type: text/plain; charset=utf-8
                     Accept-Ranges: bytes
   ```

   In the following example, the system displays the public_readOnly' container details, on the admin
   account:

   ```
   # swift stat public_readOnly -v --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
   --os-project-name admin --os-project-domain-name Default --os-username admin
   --os-user-domain-name Default --os-password Passw0rd --auth-version 3
   URL: http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
   /public_readOnly
          Auth Token: 957d6c37155b44d3a476441bc927835d
             Account: AUTH_bea5a0c632e54eaf85e9150a16c443ce
           Container: public_readOnly
             Objects: 0
               Bytes: 0
            Read ACL:
           Write ACL:
             Sync To:
            Sync Key:
       Accept-Ranges: bytes
      X-Storage-Policy: Policy-0
         X-Timestamp: 1432795292.10297
          X-Trans-Id: tx9b05c2135a9c4034b910c-0055677dad
        Content-Type: text/plain; charset=utf-8
   ```

   By default, only users who are having a Keystone role specified in the proxy-server.conf
   operator_roles option are allowed to create container on an account.

To list `operator_roles` on the IBM Spectrum Scale system during installation, issue the **mmobj config list** command as shown in the following example:

```
mmobj config list --ccrfile proxy-server.conf --section filter:keystoneauth --property operator_roles
```

To list `operator_roles` in all other cases, issue the **mmobj config list** with the following parameters:

```
mmobj config list --ccrfile proxy-server.conf --section filter:keystone --property operator_roles
```

Keystone administrator can also use the container to control access to the objects by using an access control list (ACL). The following example shows that when a member of the `admin` account tries to display the details of `public_readOnly` account, the process fails because it does not have an operator role or access control defined:

```
# swift stat public_readOnly -v --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name admin --os-project-domain-name Default --os-username member
--os-user-domain-name Default --os-password Passw0rd --auth-version 3
Container HEAD failed: http://tully-ces-ip.adcons.spectrum:8080/v1
/AUTH_bea5a0c632e54eaf85e9150a16c443ce/public_readOnly 403 Forbidden
```

Related tasks:

"Creating read ACLs to authorize object users"
The Keystone administrator can create container ACLs to grant read permissions using `X-Container-Read` headers in curl tool or `–read-acl flag` in the Swift Command Line Client.

"Creating write ACLs to authorize object users" on page 330
The Keystone administrator can create container ACLs to grant write permissions using X-Container-Write headers in the curl tool or `–write-acl flag` in the Swift Command Line Client.

Creating read ACLs to authorize object users:

The Keystone administrator can create container ACLs to grant read permissions using `X-Container-Read` headers in curl tool or `–read-acl flag` in the Swift Command Line Client.

The following example shows how to create read permission in an ACL.

1. Upload the object *imageA.JPG* to *public_readOnly* container as the Keystone administrator.

```
# swift upload public_readOnly imageA.JPG --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name admin --os-project-domain-name Default --os-username admin
--os-user-domain-name Default --os-password Passw0rd --auth-version 3
imageA.JPG
```

2. Issue the **swift post** command to provide public read access to the *public_readOnly* container.

```
# swift post public_readOnly --read-acl ".r:*,.rlistings" --os-auth-url http://tully-ces-ip.
adcons.spectrum:35357/v3 --os-project-name admin --os-project-domain-name Default
--os-username admin --os-user-domain-name Default --os-password Passw0rd --auth-version 3
```

Note: The `.r:*` ACL specifies access for any referrer regardless of account affiliation or user name. The `.rlistings` ACL allows to list the containers and read (download) objects.

3. Issue the **swift stat** command at the container level to see the access details.

```
# swift stat public_readOnly -v --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name admin --os-project-domain-name Default --os-username admin
--os-user-domain-name Default --os-password Passw0rd --auth-version 3
            URL: http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/public_readOnly
      Auth Token: 91a27a5ed8dc40d582e71844ca019c32
         Account: AUTH_bea5a0c632e54eaf85e9150a16c443ce
       Container: public_readOnly
         Objects: 3
           Bytes: 8167789
        Read ACL: .r:*,.rlistings
       Write ACL:
         Sync To:
        Sync Key:
   Accept-Ranges: bytes
```

```
          X-Trans-Id: tx73b0696705b94bf885bd5-0055678ab1
    X-Storage-Policy: Policy-0
        X-Timestamp: 1432795292.10297
       Content-Type: text/plain; charset=utf-8
```

4. As the *student* user from the *students* account, list and download the details of *public_readOnly* container that is created in the *admin* account.

Listing the details:

```
# swift stat public_readOnly -v --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name students --os-project-domain-name Default --os-username student1
--os-user-domain-name Default --os-password Passw0rd --auth-version 3 --os-storage-url
http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
              URL: http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/public_readOnly
       Auth Token: d6ee0fb5e33748b1b9035a3b690c7587
          Account: AUTH_bea5a0c632e54eaf85e9150a16c443ce
        Container: public_readOnly
          Objects: 3
            Bytes: 8167789
         Read ACL:
        Write ACL:
          Sync To:
         Sync Key:
    Accept-Ranges: bytes
 X-Storage-Policy: Policy-0
      X-Timestamp: 1432795292.10297
       X-Trans-Id: tx09893920a6154faab6ace-0055678f6d
     Content-Type: text/plain; charset=utf-8
```

Listing the container objects:

```
# swift list public_readOnly --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name students --os-project-domain-name Default --os-username student1
--os-user-domain-name Default --os-password Passw0rd --auth-version 3 --os-storage-url
http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
file.txt
imageA.JPG
imageB.JPG
```

Downloading container objects:

```
# swift download public_readOnly --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name students --os-project-domain-name Default --os-username student1
--os-user-domain-name Default --os-password Passw0rd --auth-version 3 --os-storage-url
http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
imageB.JPG [auth 0.321s, headers 0.380s, total 0.390s, 37.742 MB/s]
file.txt [auth 0.533s, headers 0.594s, total 0.594s, 0.000 MB/s]
imageA.JPG [auth 0.119s, headers 0.179s, total 18.135s, 0.308 MB/s]
```

5. As the *student1* user from the *students* account, receive deny write access, while trying to upload new content in the *public_readOnly* container:

```
# swift upload public_readOnly photo.jpg --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name students --os-project-domain-name Default --os-username student1
--os-user-domain-name Default --os-password Passw0rd --auth-version 3 --os-storage-url
http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
Warning: failed to create container 'public_readOnly': 403 Forbidden:

Forbidden

Access was denied to this resourc
Object PUT failed: http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/public_readOnly/photo.jpg 403 Forbidden
```

Manipulating the read ACLs

The following table list different read ACLs combinations:

Table 38. ACL options that are available to manipulate object read ACLs

Permission	Read ACL options
Read for all referrers	.r:*
Read and list for all referrers and listing	.r:*,.rlistings
Read and list for a user in a specific project	<project_name|project_id>:<user_name|user_id>
Read and list for a user in every project	*:<user_name|user_id>
Read and list for every user in a project	<project_name|project_id>:<*>
Read and list for every user in every project	<*>:<*>

Note: Use comma (,) to separate ACLs. For example, –read-acl `admin:admin,students:student1`.

Related tasks:

"Creating containers" on page 326
The Object Storage organizes data in account, container, and object. Each account and container is an individual database that is distributed across the cluster. An account database contains the list of containers in that account. A container database contains the list of objects in that container.

"Creating write ACLs to authorize object users"
The Keystone administrator can create container ACLs to grant write permissions using X-Container-Write headers in the curl tool or –write-acl flag in the Swift Command Line Client.

Creating write ACLs to authorize object users:

The Keystone administrator can create container ACLs to grant write permissions using X-Container-Write headers in the curl tool or –write-acl flag in the Swift Command Line Client.

Provides an example on how to configure write ACLs by using curl tool.

1. Create token and proceed to create a container named *writeOnly* with write permissions for *member* user who is part of the *admin* project and *student1* user who is part of the *students* project.

```
token=$(openstack --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name admin --os-project-domain-name Default --os-username admin
--os-user-domain-name Default --os-password Passw0rd --os-identity-api-version 3
token issue | grep -w "id" | awk '{print $4}')

# curl -i http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/writeOnly -X PUT -H "Content-Length: 0" -H "X-Auth-Token: ${token}" -H
"X-Container-Write: admin:member,students:student1" -H "X-Container-Read: "
HTTP/1.1 201 Created
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txf7b0bfef877345949c61c-005567b9d1
Date: Fri, 29 May 2015 00:58:57 GMT
```

2. Issue a token as *student1* from the *students* project and upload an object by using the curl tool.

```
token=$(openstack --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name students --os-project-domain-name Default --os-username student1
--os-user-domain-name Default --os-password Passw0rd --os-identity-api-version 3
token issue | grep -w "id" | awk '{print $4}')

# curl -i http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/writeOnly/imageA.JPG -X PUT -H "X-Auth-Token: ${token}"  --upload-file  imageA.JPG
HTTP/1.1 100 Continue
HTTP/1.1 201 Created
Last-Modified: Fri, 29 May 2015 01:11:28 GMT
```

```
Content-Length: 0
Etag: 95d8c44b757f5b0c111750694dffef2b
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx6caa0570bfcd419782274-005567bcbe
Date: Fri, 29 May 2015 01:11:28 GMT
```

3. List the state of the *writeOnly* container as *student1* user of the *students* project. This operation fails as the user does not have the required privileges.

```
# curl -i http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/writeOnly/imageA.JPG -X HEAD  -H "X-Auth-Token: ${token}"
HTTP/1.1 403 Forbidden
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx4f7dfbfd74204785b6b50-005567bd8c
Content-Length: 0
Date: Fri, 29 May 2015 01:14:52 GMT
```

4. Grant read permissions to *student1* user of the *students* project:

```
token=$(openstack --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name admin --os-project-domain-name Default --os-username admin
--os-user-domain-name Default --os-password Passw0rd --os-identity-api-version 3
token issue | grep -w "id" | awk '{print $4}')

# curl -i http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_
bea5a0c632e54eaf85e9150a16c443ce
/writeOnly -X POST -H "Content-Length: 0" -H "X-Auth-Token:
${token}" -H "X-Container-Read: students:student1"
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx77aafe0184da4b68a7756-005567beac
Date: Fri, 29 May 2015 01:19:40 GMT
```

5. Verify whether the *sutdent1* user has the read access now.

```
token=$(openstack --os-auth-url http://tully-ces-ip.adcons.spectrum:35357/v3
--os-project-name students --os-project-domain-name Default --os-username student1
--os-user-domain-name Default --os-password Passw0rd --os-identity-api-version 3
token issue | grep -w "id" | awk '{print $4}')

# curl -i http://tully-ces-ip.adcons.spectrum:8080/v1/AUTH_bea5a0c632e54eaf85e9150a16c443ce
/writeOnly -X GET  -H "X-Auth-Token: ${token}"
HTTP/1.1 200 OK
Content-Length: 11
X-Container-Object-Count: 1
Accept-Ranges: bytes
X-Storage-Policy: Policy-0
X-Container-Bytes-Used: 5552466
X-Timestamp: 1432861137.91693
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx246b39018a5c4bcb90c7f-005567bff3
Date: Fri, 29 May 2015 01:25:07 GMT

imageA.JPG
```

Note: Object Storage does not support public write ACLs.

Related tasks:

"Creating containers" on page 326
The Object Storage organizes data in account, container, and object. Each account and container is an individual database that is distributed across the cluster. An account database contains the list of containers in that account. A container database contains the list of objects in that container.

"Creating read ACLs to authorize object users" on page 328
The Keystone administrator can create container ACLs to grant read permissions using `X-Container-Read` headers in curl tool or `--read-acl` flag in the Swift Command Line Client.

Authorization limitations

Authorization limitations are specific to the protocols that are used to access data.

NFS ACL limitations

ACLs are stored as NFSv4 ACLs in the file system.

For more information on the known limitations of the NFSV4 ACLs, see "Exceptions and limitations to NFS V4 ACLs support" on page 336.

SMB ACL limitations

The following are the SMB ACL limitations:
- ACL of a new child file or directory depends on the ACL type, the file system settings, and the ACL of the parent directory. Depending on these variables, the results in the IBM Spectrum Scale might be slightly different than in Microsoft Windows. For example, if the parent directory is set to have two ACEs, for example full access for owner and for everyone, the Windows default is to create two ACLs for the child. One is to allow full access for owner and other to allow full access for everyone. The IBM Spectrum Scale system by default creates six ACLs to allow and deny ACLs for owner, group, and everyone.
- If domain server manages the UID and GID mapping, the UID and GID mappings must be configured in the domain server before an ACE for that user or group can be created.
- Users and groups that belonged to another domain, and was migrated to a new domain by using the SID-History mechanism, cannot be stored in an ACL.
- Most well-known SIDs and built-in SIDs cannot be stored in an ACL. Only the "Everyone" SID can be stored and used in an IBM Spectrum Scale system.
- The SMB ACLs cannot be modified when LDAP-based authentication is used for file access.
- Microsoft Windows enables you to limit the scope of inheritance for an ACE to one inheritance by selecting the **Apply these permissions to objects and/or containers within this container only** check box in the Windows Explorer. The IBM Spectrum Scale system does not support to configure this option and limit the scope of inheritance for an ACL.
- ACL inheritance stops at fileset junction points. New filesets always have the default ACL (770 root root).
- The root path of every SMB share needs read permission (read data, read attribute, read extended attribute) for everyone, to prevent the unexpected behavior of, for example, Windows Explorer.
- To prevent display of Access Denied errors, the user must have the read attribute permission on all parent directories, when they have access to a file or directory.
- The value of the dacl_protected bit related to the Include Inheritable permissions from this object's parent check box can be changed only through SMB. The ACL commands cannot access this field. Setting a new ACL resets this field.
- The commands that are used to work on the ACLs do not support recursive updates of inherited ACEs in the file tree.
- Access privileges defined in Windows are not honored. Those privileges are tied to administrator groups and allow access, where the ACL alone does not grant it.
- Audit and alarm ACEs are not supported inside an ACL.
- The Bypass Traverse Check is implemented in GPFS for SMB clients only. Clients that use other protocols might still be locked out because the parent tree of an export has more restrictive ACLs than the export itself.
- POSIX-style ACLs are not supported.
- Similar to the POSIX standard, to read the content of a subdirectory, apart from the read permission in the ACL of this subdirectory, the user also need to have traversal permission (SEARCH in Windows,

EXECUTE in POSIX) for all of the parent directories. You can set the traverse permission in the "Everyone" group ACE at the share root, and inherit this privilege to all subdirectories. For the SMB protocol, this is applicable only if the configuration option *bypassTraversalCheck* is disabled.

• Even though the underlying file system does not enforce the permissions for extended attributes (READ_NAMED and WRITE_NAMED), these are enforced for SMB clients.

ACL limitations that are applicable to all protocols

The following limitations are applicable to all protocols:

• When creating a file system, the user needs to specify -k nfs4 to specifically use NFSv4 ACLs, otherwise the default -k all uses both POSIX ACLs and NFSV4 ACLs.

• The IBM Spectrum Scale Object Storage does not do file share with NFS and SMB.

Chapter 23. Considerations for GPFS applications

Application design must consider the exceptions to the Open Group technical standards for the **stat()** system call and NFS V4 ACLs. Also, a technique to check if a file system is controlled by GPFS has been provided.

Consider the following:
- "Exceptions to Open Group technical standards"
- "Determining if a file system is controlled by GPFS"
- "Exceptions and limitations to NFS V4 ACLs support" on page 336

Exceptions to Open Group technical standards

GPFS is designed in a way that most applications written to the Open Group technical standard for file system calls can access the GPFS data without any modification. However, there are some exceptions.

Applications that depend on exact reporting of changes to the following fields returned by the **stat()** call may not work as expected:

1. **exact mtime**
2. **mtime**
3. **ctime**
4. **atime**

Providing exact support for these fields would require significant performance degradation to all applications executing on the system. These fields are guaranteed accurate when the file is closed.

These values will be accurate on a node right after it accesses or modifies a file, but may not be accurate for a short while when a file is accessed or modified on some other node.

If 'exact mtime' is specified for a file system (using the **mmcrfs** or **mmchfs** commands with the **-E yes** flag), the **mtime** and **ctime** values are always correct by the time the **stat()** call gives its answer. If 'exact mtime' is not specified, these values will be accurate after a couple of minutes, to allow the synchronization daemons to propagate the values to all nodes. Regardless of whether 'exact mtime' is specified, the **atime** value will be accurate after a couple of minutes, to allow for all the synchronization daemons to propagate changes.

Alternatively, you may use the GPFS calls, **gpfs_stat()** and **gpfs_fstat()** to return exact **mtime** and **atime** values.

The delayed update of the information returned by the **stat()** call also impacts system commands which display disk usage, such as **du** or **df**. The data reported by such commands may not reflect changes that have occurred since the last sync of the file system. For a parallel file system, a sync does not occur until all nodes have individually synchronized their data. On a system with no activity, the correct values will be displayed after the sync daemon has run on all nodes.

Determining if a file system is controlled by GPFS

A file system is controlled by GPFS if the **f_type** field in the **statfs** structure returned from a **statfs()** or **fstatfs()** call has the value 0x47504653, which is GPFS in ASCII.

This constant is in the **gpfs.h** file, with the name **GPFS_SUPER_MAGIC**. If an application includes **gpfs.h**, it can compare **f_type** to **GPFS_SUPER_MAGIC** to determine if the file system is controlled by GPFS.

Exceptions and limitations to NFS V4 ACLs support

Review the exceptions and limitations to NFS V4 ACLs in IBM Spectrum Scale.

1. IBM Spectrum Scale has limited support for ACLs, but only with Samba on Linux. In that environment, IBM Spectrum Scale can only save and retrieve Alarm and Audit access control entries (ACEs). No actions are defined that can be taken for ACEs during ACL evaluation.

2. Some types of access for which NFS V4 defines controls do not currently exist in IBM Spectrum Scale. For these, ACL entries are accepted and saved, but because there is no corresponding operation they have no effect. These include `READ_NAMED`, `WRITE_NAMED`, and `SYNCHRONIZE`.

   Note: Even if IBM Spectrum Scale ignores these bits, the SMB service enforces them on the protocol level.

3. AIX requires that `READ_ACL` and `WRITE_ACL` always be granted to the object owner. Although this contradicts NFS Version 4 protocol, it is viewed that this is an area where users would otherwise erroneously leave an ACL that only privileged users could change. Because ACLs are themselves file attributes, `READ_ATTR` and `WRITE_ATTR` are similarly granted to the owner. Because it would not make sense to then prevent the owner from accessing the ACL from a non-AIX node, IBM Spectrum Scale has implemented this exception everywhere.

4. AIX does not support the use of special name values other than `owner@`, `group@`, and `everyone@`. Therefore, these are the only valid special name values for use in IBM Spectrum Scale NFS V4 ACLs.

5. NFS V4 allows ACL entries that grant permission to users or groups to change the ownership of a file with a command such as the `chown` command. For security reasons, IBM Spectrum Scale now restricts these permissions so that a nonprivileged user can `chown` such a file only to himself or to a group that he or she is a member of.

6. With some limitations, Windows clients that access IBM Spectrum Scale via Samba can use their native NFS ACLs, which are mapped to the underlying NFS v4 ACLs. For limitations, see "Authorization limitations" on page 332.

7. Ganesha supports NFS v4 ACLs to and from IBM Spectrum Scale. However, to export a file system with cNFS/KNFS, you must configure the file system to support POSIX ACLs. Use the `mmcrfs` command with the `-k all` or `-k posix` parameter. With Samba, use the `-k nfs4` parameter. NFS V4 Linux servers handle ACLs properly only if they are stored in GPFS as POSIX ACLs. For more information, see "Linux ACLs and extended attributes."

8. The cluster can include Samba, CES NFS, AIX NFS, and IBM Spectrum Scale Windows nodes.

9. NFS V4 ACLs can be stored in GPFS file systems using Samba exports, NFS V4 AIX servers, GPFS Windows nodes, `aclput`, and `mmputacl`. Clients of Linux V4 servers cannot see stored ACLs but can see the permissions from the mode.

For more information about ACLs and NFS export, see Chapter 22, "Managing GPFS access control lists," on page 303.

Linux ACLs and extended attributes

NFS V4 uses the existing POSIX ACLs and the extended attribute support in Linux that is supported by GPFS.

Note: This topic applies only to cNFS/kNFS.

Although the NFS V4 protocol defines a richer ACL model similar to Windows ACLs, the Linux implementation maps those ACLs to POSIX ACLs before passing them to the underlying file system. This

mapping is done in nfsd. This means that if you set an ACL from a client and then fetch it back, you will see that what the server returns is not exactly what you set. This is because the ACL you set was converted to a POSIX ACL and then back again.

NFS V4 ACLs are more fine-grained than POSIX ACLs, so the POSIX-to-NFS V4 translation is close to perfect, but the NFS V4-to-POSIX translation is not. The NFS V4 server attempts to err on the side of mapping to a stricter ACL. There is a very small set of NFS V4 ACLs that the server rejects completely (for example, any ACL that attempts to explicitly DENY rights to read attributes), but otherwise, the server tries very hard to accept ACLs and map them as best it can.

ACLs that are set through AIX/NFS V4 and Windows nodes are written as NFS V4 ACLs to GPFS, whereas ACLs that are set through Linux/NFS V4 are written as POSIX ACLs to GPFS. Currently, GPFS does not provide an interface to convert on-disk NFS V4 ACLs to POSIX ACLs. This means that if ACLs are written through either AIX/NFS V4 or Windows, they cannot be read by Linux/NFS V4. In this case, a Linux NFS V4 server constructs an ACL from the permission mode bits only and ignores the ACL on the file.

General CES NFS Linux exceptions and limitations

GPFS has some CES NFS Linux exceptions and limitations.

Limitations with Microsoft Windows

NFS clients are not supported on Microsoft Windows.

IBM CES NFS stack limitations

CES NFS limitations are described here:

- Changes to the IBM® CES NFS global configuration are not dynamic. NFS services will automatically restart during the execution of the `mmnfs export load` and `mmnfs config change` commands. During this time, an NFS client with a soft mount might lose connectivity. This might result in an application failure on the client node. An NFS client with a hard mount might "stall" during the NFS restart.
- Whenever NFS is restarted, a grace period will ensue. The NFS grace period is user configurable, and the default NFS grace period is 60 seconds. If NFS global configuration changes are performed sequentially, then NFS services will be restarted multiple times, leading to a cumulative extended grace period. This might prevent NFS clients from reclaiming their locks, possibly leading to an application failure on the client node.
- Exporting symbolic links is not supported in CES NFS.

Considerations for the use of direct I/O (O_DIRECT)

When a file is opened in the `O_DIRECT` mode (direct I/O mode), GPFS transfers data directly between the user buffer and the file on the disk.

Using direct I/O may provide some performance benefits in the following cases:
- The file is accessed at random locations.
- There is no access locality.

Direct transfer between the user buffer and the disk can only happen if all of the following conditions are true:
- The number of bytes transferred is a multiple of 512 bytes.
- The file offset is a multiple of 512 bytes.
- The user memory buffer address is aligned on a 512-byte boundary.

When these conditions are *not* all true, the operation will still proceed but will be treated more like other normal file I/O, with the **O_SYNC** flag that flushes the dirty buffer to disk.

When these conditions *are* all true, the GPFS page pool is not used because the data is transferred directly; therefore, an environment in which most of the I/O volume is due to direct I/O (such as in databases) will not benefit from a large page pool. Note, however, that the page pool still needs to be configured with an adequate size, or left at its default value, because the page pool is also used to store file metadata (especially for the indirect blocks required for large files).

With direct I/O, the application is responsible for coordinating access to the file, and neither the overhead nor the protection provided by GPFS locking mechanisms plays a role. In particular, if two threads or nodes perform direct I/O concurrently on overlapping portions of the file, the outcome is undefined. For example, when multiple writes are made to the same file offsets, it is undetermined which of the writes will be on the file when all I/O is completed. In addition, if the file has data replication, it is not guaranteed that all the data replicas will contain the data from the same writer. That is, the contents of each of the replicas could diverge.

Even when the I/O requests are aligned as previously listed, in the following cases GPFS will not transfer the data directly and will revert to the slower buffered behavior:
- The write causes the file to increase in size.
- The write is in a region of the file that has been preallocated (via **gpfs_prealloc()**) but has not yet been written.
- The write is in a region of the file where a "hole" is present; that is, the file is sparse and has some unallocated regions.

When direct I/O requests are aligned but none of the previously listed conditions (that would cause the buffered I/O path to be taken) are present, handling is optimized this way: the request is completely handled in kernel mode by the GPFS kernel module, without the GPFS daemon getting involved. Any of the following conditions, however, will still result in the request going through the daemon:
- The I/O operation needs to be served by an NSD server.
- The file system has data replication. In the case of a write operation, the GPFS daemon is involved to produce the log records that ensure that the replica contents are identical (in case of a failure while writing the replicas to disk).
- The operation is performed on the Windows operating system.

Note that setting the **O_DIRECT** flag on an open file with **fcntl (fd, F_SETFL,[..])**, which may be allowed on Linux, is ignored in a GPFS file system.

Because of a limitation in Linux, I/O operations with **O_DIRECT** should not be issued concurrently with a **fork(2)** system call that is invoked by the same process. Any calls to **fork()** in the program should be issued only after **O_DIRECT** I/O operations are completed. That is, **fork()** should not be invoked while **O_DIRECT** I/O operations are still pending completion. For more information, see the **open(2)** system call in the Linux documentation.

NFS protocol node limitations

NFS protocol nodes have conflicting NLM locking services between the Linux kernel and CES NFS.

When mounting an NFSv3 file system on a protocol node, the Linux kernel **lockd** daemon registers with the rpcbind, preventing the CES NFS lock service from taking effect. If you need to mount an NFSv3 file system on a CES NFS protocol node, use the **-o nolock mount** option to prevent invoking the Linux kernel **lockd** daemon.

Note: cNFS is not supported with sudo wrappers.

Chapter 24. Accessing a remote GPFS file system

GPFS allows users shared access to files in either the cluster where the file system was created, or other GPFS clusters. File system access by the cluster where the file system was created is implicit.

The ability to access and mount GPFS file systems owned by other clusters in a network of sufficient bandwidth is accomplished using the **mmauth**, **mmremotecluster** and **mmremotefs** commands. Each site in the network is managed as a separate cluster, while allowing shared file system access.

The cluster owning the file system is responsible for administering the file system and granting access to other clusters on a per cluster basis. After access to a particular file system has been granted to nodes in another GPFS cluster, the nodes can mount the file system and perform data operations as if the file system were locally owned.

Each node in the GPFS cluster requiring access to another cluster's file system must be able to open a TCP/IP connection to every node in the other cluster.

Nodes in two separate remote clusters mounting the same file system are not required to be able to open a TCP/IP connection to each other. For example, if a node in `clusterA` mounts a file system from `clusterB`, and a node in `clusterC` desires to mount the same file system, nodes in `clusterA` and `clusterC` do not have to communicate with each other.

Each node in the GPFS cluster requiring file system access must have one of the following:
* A virtual connection to the file system data through an NSD server (refer to Figure 8 on page 340).
* A physical connection to the disks containing file system data (refer to Figure 9 on page 340).

In this example, network connectivity is required from the nodes in `clusterB` to all the nodes in `clusterA` even if the nodes in `clusterB` can access the disks in `clusterA` directly.

Note: Even when remote nodes have direct connectivity to the SAN, they will still use a connection through an NSD server for any NSDs that have been configured with Persistent Reserve (PR). If you want the remote nodes to access the disks through their direct connection to the SAN, you must ensure that PR is not enabled on the NSDs. See "Enabling and disabling Persistent Reserve" on page 166.
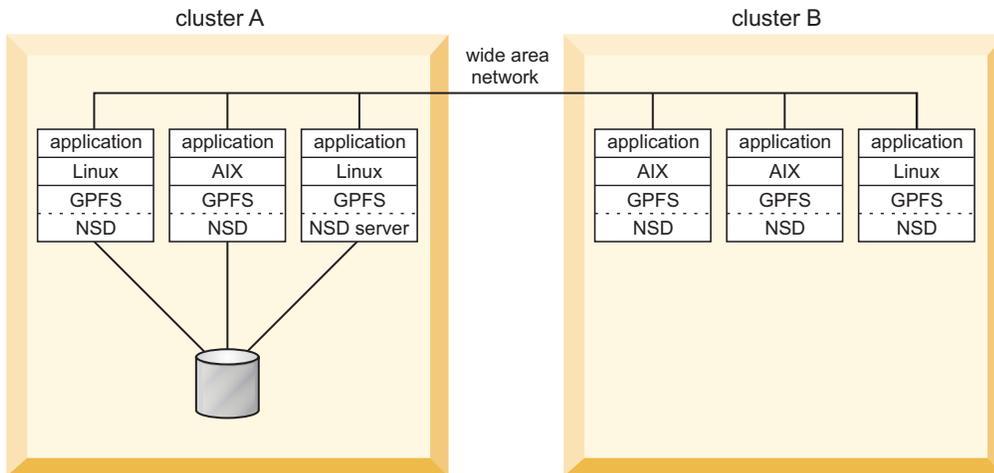
*Figure 8. Remote mount of a file system using NSD server access*
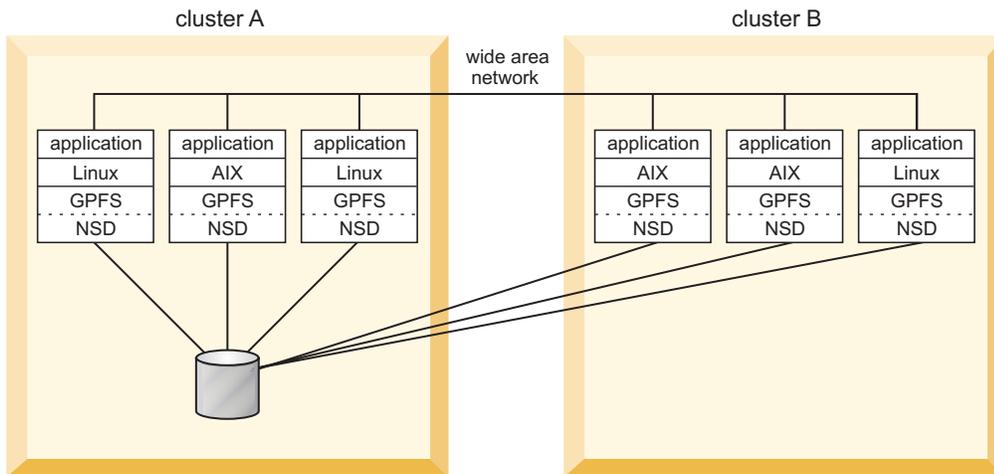
Figure 9. Remote mount of a file system using SAN-attached disks

Figure 10 on page 341 illustrates a multi-cluster configuration with multiple NSD servers. In this configuration:

- The two nodes in Cluster 1 are defined as the NSD servers (you can have up to eight NSD server nodes).
- All three clusters are connected with Gigabit Ethernet.
- Cluster 1 shares an InfiniBand switch network with Cluster 2 and an InfiniBand switch network with Cluster 3.

Note: Protocol support is not available in a multi-cluster configuration.

In order to take advantage of the fast networks and to use the nodes in Cluster 1 as NSD servers for Cluster 2 and Cluster 3, you must configure a subnet for each of the supported clusters. For example issuing the command:

- **mmchconfig subnets="<*IB_Network_1*> <*IB_Network_1*>/Cluster1"** in Cluster 2 allows nodes N_2 through N_x to use N_1 as an NSD server with InfiniBand Network 1 providing the path to the data.
- **mmchconfig subnets="<*IB_Network_2*> <*IB_Network_2*>/Cluster1"** in Cluster 3 allows nodes N_{2+x} through N_{y+x} to use N_{1+x} as an NSD server with InfiniBand Network 2 providing the path to the data.
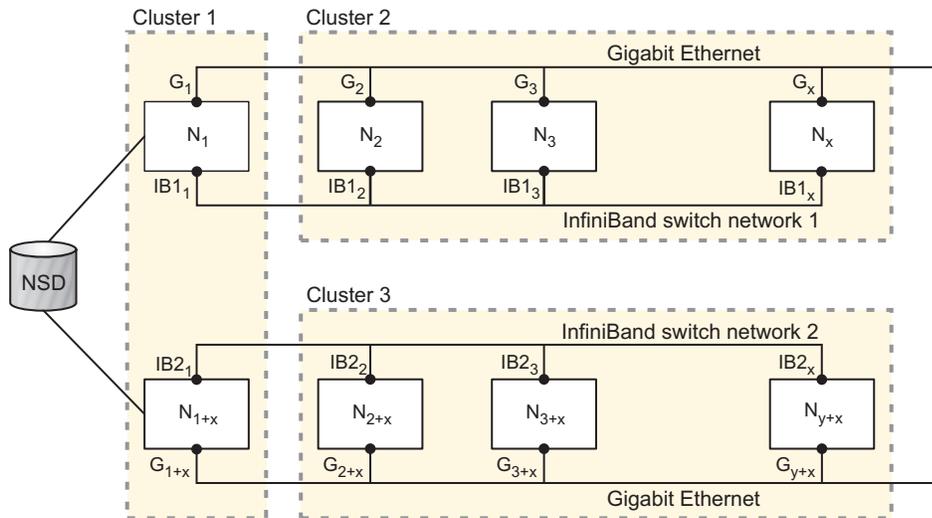
Figure 10. Multi-cluster configuration with multiple NSD servers

When you implement file access from other clusters, consider these topics:

- "Remote user access to a GPFS file system"

Remote user access to a GPFS file system

In a cluster environment that has a single user identity namespace, all nodes have user accounts set up in a uniform manner. This is usually accomplished by having equivalent **/etc/passwd** and **/etc/group** files on all nodes in the cluster.

For consistency of ownership and access control, a uniform user identity namespace is preferred. For example, if user Jane Doe has an account on nodeA with the user name **janedoe** and user ID **1001** and group ID **500**, on all other nodes in the same cluster Jane Doe will have an account with the same user and group IDs. GPFS relies on this behavior to perform file ownership and access control tasks.

If a GPFS file system is being accessed from a node belonging to another GPFS cluster, the assumption about the uniform user account infrastructure might no longer be valid. Since different clusters can be administered by different organizations, it is possible for each of the clusters to have a unique set of user accounts. This presents the problem of how to permit users to access files in a file system owned and served by another GPFS cluster. In order to have such access, the user must be somehow known to the other cluster. This is usually accomplished by creating a user account in the other cluster, and giving this account the same set of user and group IDs that the account has in the cluster where the file system was created.

To continue with this example, Jane Doe would need an account with user ID **1001** and group ID **500** created in every other GPFS cluster from which remote GPFS file system access is desired. This approach is commonly used for access control in other network file systems, (for example, NFS or AFS®), but might pose problems in some situations.

For example, a problem arises if Jane Doe already has an account in some other cluster, but the user ID associated with this account is not **1001**, and another user in the other cluster has user ID **1001**. It would require a considerable effort on the part of system administrator to ensure that Jane Doe's account has the same set of IDs on all clusters. It is more desirable to be able to use the existing accounts without having to make changes. GPFS helps to solve this problem by optionally performing user ID and group ID remapping internally, using user-supplied helper applications. For a detailed description of the GPFS user ID remapping convention, see the IBM white paper entitled *UID Mapping for GPFS in a Multi-cluster Environment* in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/SSFKCN/com.ibm.cluster.gpfs.doc/gpfs_uid/uid_gpfs.html).

Access from a remote cluster by a root user presents a special case. It is often desirable to disallow root access from a remote cluster while allowing regular user access. Such a restriction is commonly known as root squash. A root squash option is available when making a file system available for mounting by other clusters using the **mmauth** command. This option is similar to the NFS root squash option. When enabled, it causes GPFS to squash superuser authority on accesses to the affected file system on nodes in remote clusters.

This is accomplished by remapping the credentials: user id (UID) and group id (GID) of the root user, to a UID and GID specified by the system administrator on the home cluster, for example, the UID and GID of the user nobody. In effect, root squashing makes the root user on remote nodes access the file system as a non-privileged user.

Although enabling root squash is similar to setting up UID remapping, there are two important differences:
1. While enabling UID remapping on remote nodes is an option available to the remote system administrator, root squashing need only be enabled on the local cluster, and it will be enforced on remote nodes. Regular UID remapping is a user convenience feature, while root squashing is a security feature.
2. While UID remapping requires having an external infrastructure for mapping between local names and globally unique names, no such infrastructure is necessary for enabling root squashing.

When both UID remapping and root squashing are enabled, root squashing overrides the normal UID remapping mechanism for the root user.

Using NFS/SMB protocol over remote cluster mounts

IBM Spectrum Scale allows you to create NFS and SMB exports on remotely mounted file systems.

The following diagram shows the high-level flow of this feature.
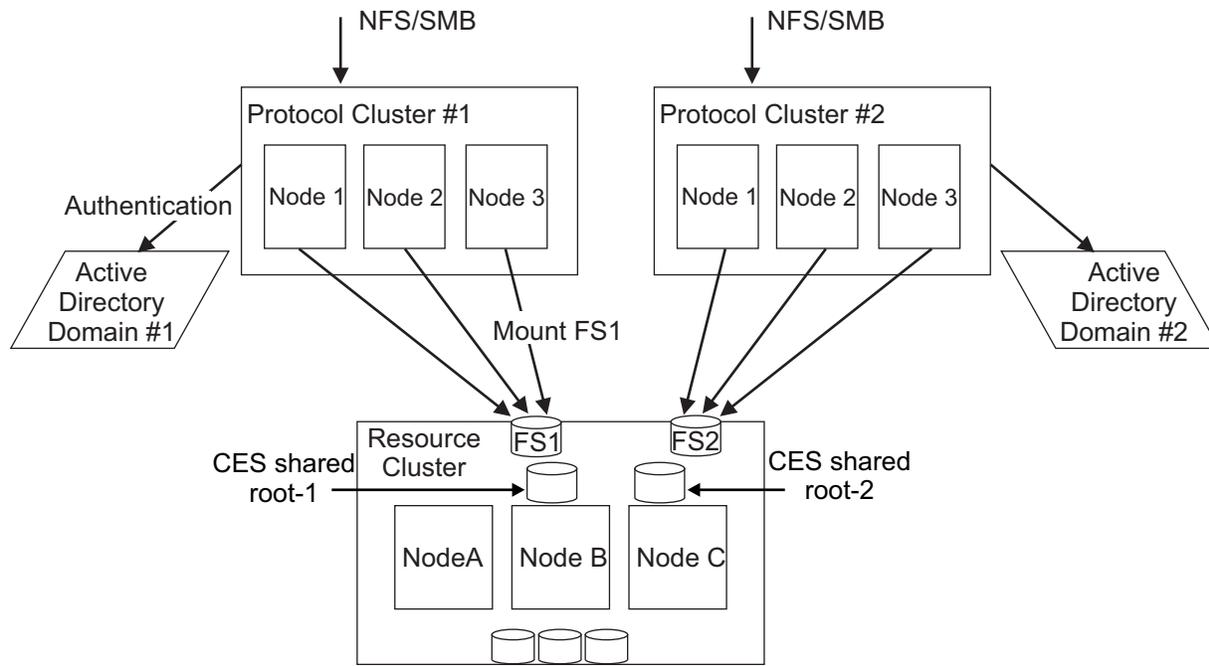
Figure 11. High-level flow of protocols on remotely mounted file systems

This allows you to separate the tasks performed by each cluster. Storage cluster owns the file systems and the storage. Protocol clusters contain the protocol node that provides access to the remotely mounted file system through NFS or SMB. In this configuration, each cluster is managed independently. For more information, see "Important information about remote access" on page 353.

Here, the storage cluster owns a file system and the protocol cluster remotely mounts the file system. The protocol nodes (CES nodes) in the protocol cluster export the file system via SMB and NFS.

You can define one set of protocol nodes per cluster, using multiple independent protocol clusters which remotely mount file systems. Protocol clusters can share access to a storage cluster but not to a file system. Each protocol cluster requires a dedicated file system. Each protocol cluster can have a different authentication configuration, thus allowing different authentication domains while keeping the data at a central location. Another benefit is the ability to access existing ESS-based file systems through NFS or SMB without adding nodes to the ESS cluster.

Configuring protocols on a separate cluster

The process for configuring Cluster Export Services (CES) in a multi-cluster environment in many respects is the same as for a single cluster, however, there are a few differences mainly in the procedure order.

This procedure assumes an environment with the server, network, storage, and operating systems are installed and ready for IBM Spectrum Scale. For more information, see *Installing IBM Spectrum Scale on Linux nodes and deploying protocols* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Do the following steps:

1. Install IBM Spectrum Scale on all nodes that are in the storage and protocol clusters.

Note: If you install a protocol cluster into an environment with an existing IBM Spectrum Scale cluster, the IBM Spectrum Scale version used should comply with the protocol cluster. The installation can be performed either manually or by using the installation toolkit. Do not create clusters or file systems or Cluster Export Services yet.

2. Create the storage and protocol clusters.

   Note: Proceed with cluster creation of the storage cluster and one or more protocol clusters. Ensure that the configuration parameter maxBlockSize is set to the same value on all clusters.

3. Create file systems on the storage cluster, taking the following into consideration:
   - CES shared root file system – Each protocol cluster requires its own CES shared root file system. Having a shared root file system that is different from the file system that serves data eases the management of CES.
   - Data file systems – At least one file system is required for each protocol cluster configured for Cluster Export Services. A data file system can only be exported from a single protocol cluster.

4. Before installing and configuring Cluster Export Services, consider the following points:
   - Authentication - Separate authentication schemes are supported for each CES cluster.
   - ID mapping - The ID mapping of users authenticating to each CES cluster. It is recommended to have unique ID mapping across clusters, but not mandatory.

     Note: You must judiciously determine the ID mapping requirements and prevent possible interference or security issues.
   - GUI - GUI support for remote clusters is limited. Each cluster should have its own GUI. The GUI may be installed onto CES nodes but performance must be taken into consideration.
   - Object - Object is not supported in multi-cluster configurations.
   - For a list of limitations, see *Limitations of protocols on remotely mounted file systems* in *IBM Spectrum Scale: Administration Guide*.

5. Configure clusters for remote mount. For more information, see *Mounting a remote GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

6. Install and configure Cluster Export Services by using the installation toolkit or manually. For more information, see *Installing IBM Spectrum Scale on Linux nodes with the installation toolkit* and *Manually installing the IBM Spectrum Scale software packages on Linux nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

   Note: Use the remotely mounted CES shared root file system.

7. Once SMB and/or NFS is enabled, new exports can be created on the remotely mounted data file system.

Managing multi-cluster protocol environments

In multi-cluster protocol environments, each cluster is managed independently, and there is no central management for all clusters included in IBM Spectrum Scale. You can centrally manage multiple IBM Spectrum Scale clusters by using Spectrum Control.

Consider the following aspects while you manage a multi-cluster protocol environment:
- Each cluster requires its own GUI. The GUI might be installed onto the CES nodes but performance must be taken into consideration.
- Each cluster has its own Rest API.
- Each cluster has its own health monitoring. This means that error events that are raised in the storage cluster are not visible in the protocol cluster and vice versa.
- Availability of certain performance metrics depends on the role of the cluster. That is, NFS metrics are available on protocol clusters only.
- Each cluster is installed and upgraded independently.

Due to the separation of duties (storage clusters own the file systems and protocol clusters own the NFS/SMB exports) certain management tasks must be done in the corresponding cluster:

- File system-related operations like creating file systems, filesets, or snapshots must be done in the storage cluster.
- Export-related operations like creating exports, managing CES IP addresses, and managing authentication must be done in the protocol cluster.

Since the resource cluster is unaware of the authentication setup and UID mapping, all actions that require a user or a group name must be done in the corresponding protocol cluster (for example, generate quota reports, manage ACLs).

Upgrading multi-cluster environments

There is no special process to upgrade clusters in a multi-cluster environment. When you choose a IBM Spectrum Scale version, the release should comply with release level limitations.

Note: Upgrades are performed on a cluster-boundary basis.

Once all clusters in the environment are upgraded, the release and the file system version should be changed. The release version might be changed concurrently. However, changing the file system version requires the file system to be unmounted. To view the differences between file system versions, see the *Listing file system attributes* topic in the *IBM Spectrum Scale: Administration Guide*.

To change the IBM Spectrum Scale release, issue the following command on each cluster:

```
mmchconfig release=LATEST
```

Note: Nodes that run an older version of IBM Spectrum Scale on the remote cluster will no longer be able to mount the file system. Command fails if any nodes running an older version are mounted at time command is issued.

To change the file system version, issue the following command for each file system on the storage cluster:

```
mmchfs <fs> -V full
```

If your requirements call for it, issue the following command:

```
mmchfs <fs> -V compat
```

This enables only backward-compatible format changes.

Limitations of protocols on remotely mounted file systems

You must consider certain restrictions when you plan on setting up a multi-cluster protocol environment.

Refer to the following points:
- You can configure one storage cluster and up to five protocol clusters (current limit).
- The storage cluster owns all of the exported IBM Spectrum Scale file systems. This means at least two file systems per protocol cluster (one CES shared root + one data file system)
- The storage cluster must not have any protocol nodes (CES must be disabled).
- The protocol clusters cannot own any IBM Spectrum Scale file systems, only remote mounts from the storage cluster are allowed.
- Any file system can be remotely mounted by exactly one protocol cluster. Sharing a file system between multiple protocol clusters might cause data inconsistencies.
- The primary use case for multi-cluster protocol is to allow multiple authentication configurations. The setup must not be used for extending the scalability of Cluster Export Services (CES) or to work around defined limitations (for example, number of SMB connections).

- This setup provides some level of isolation between the clusters, but there is no strict isolation of administrative operations, and there is no guarantee that administrators on one cluster cannot see data from another cluster. Strict isolation is guaranteed through NFS or SMB access only.
- Each protocol cluster must use a dedicated file system, it is not allowed to share a file system between multiple protocol clusters.
- Storage and protocol clusters are in the same site/location, high network latencies between them can cause problems.
- This setup cannot be used for Object or iSCSI services.
- Due to the separation of duties (resource clusters own the file systems and protocol clusters own the NFS/SMB exports), certain management task must be done in the corresponding cluster:
  - File system related operations (like creating file systems, filesets, creating snapshots) must be done in the resource cluster.
  - Export-related operations (creating exports, managing CES IPs, managing authentication and ACLs) must be done in the protocol cluster.

  Note: This also means that certain operations such as creation of fileset and snapshots do not work on the GUI.
- Cross-protocol change notifications will not work on remotely-mounted file systems. For example, if an NFS client changes a file, the system will not issue a "file change" notification to the SMB client which has asked for a notification.

Mounting a remote GPFS file system

Explore an example of how to mount a file system that is owned and served by another IBM Spectrum Scale cluster.

The package `gpfs.gskit` must be installed on all the nodes of the owning cluster and the accessing cluster. For more information, see the installation chapter for your operating system, such as *Installing GPFS on Linux node and deploying protocols* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The procedure to set up remote file system access involves the generation and exchange of authorization keys between the two clusters. In addition, the administrator of the GPFS cluster that owns the file system needs to authorize the remote clusters that are to access it, while the administrator of the GPFS cluster that seeks access to a remote file system needs to define to GPFS the remote cluster and file system whose access is desired.

Note: For more information on CES cluster setup, see "CES cluster setup" on page 459.

In this example, **owningCluster** is the cluster that owns and serves the file system to be mounted and **accessingCluster** is the cluster that accesses **owningCluster**.

Note: The following example uses AUTHONLY as the authorization setting. When you specify AUTHONLY for authentication, GPFS checks network connection authorization. However, data sent over the connection is not protected.

1. On **owningCluster**, the system administrator issues the **mmauth genkey** command to generate a public/private key pair. The key pair is placed in /var/mmfs/ssl. The public key file is id_rsa.pub.
   ```
   mmauth genkey new
   ```
2. On **owningCluster**, the system administrator enables authorization by entering the following command:
   ```
   mmauth update . -l AUTHONLY
   ```

3. The system administrator of **owningCluster** gives the file /var/mmfs/ssl/id_rsa.pub to the system administrator of **accessingCluster**. This operation requires the two administrators to coordinate their activities and must occur outside of the GPFS command environment.

   The system administrator of **accessingCluster** can rename the key file and put it in any directory of the node that he is working on, so long as he provides the correct path and file name in the **mmremotecluster add** command in Step 9. In this example, the system administrator renames the key file to owningCluster_id_rsa.pub.

4. On **accessingCluster**, the system administrator issues the **mmauth genkey** command to generate a public/private key pair. The key pair is placed in /var/mmfs/ssl. The public key file is id_rsa.pub.

   `mmauth genkey new`

5. On **accessingCluster**, the system administrator enables authorization by entering the following command:

   `mmauth update . -l AUTHONLY`

6. The system administrator of **accessingCluster** gives key file /var/mmfs/ssl/id_rsa.pub to the system administrator of **owningCluster**. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.

   The system administrator of **owningCluster**can rename the key file and put it in any directory of the node that he is working on, so long as he provides the correct path and file name in the **mmauth add** command in Step 7. In this example, the system administrator renames the key file to accessingCluster_id_rsa.pub.

7. On **owningCluster**, the system administrator issues the **mmauth add** command to authorize **accessingCluster** to mount file systems that are owned by **owningCluster** utilizing the key file that was received from the administrator of **accessingCluster**:

   `mmauth add accessingCluster -k accessingCluster_id_rsa.pub`

8. On **owningCluster**, the system administrator issues the **mmauth grant** command to authorize **accessingCluster** to mount specific file systems that are owned by **owningCluster**:

   `mmauth grant accessingCluster -f gpfs`

9. On **accessingCluster**, the system administrator must define the cluster name, contact nodes and public key for **owningCluster**:

   `mmremotecluster add owningCluster -n node1,node2,node3 -k owningCluster_id_rsa.pub`

   This command provides the system administrator of **accessingCluster** a means to locate the serving cluster and mount its file systems.

10. On **accessingCluster**, the system administrator issues one or more **mmremotefs** commands to identify the file systems in **owningCluster** that are to be accessed by nodes in **accessingCluster**:

    `mmremotefs add mygpfs -f gpfs -C owningCluster -T /mygpfs`

    where:

    mygpfs Is the device name under which the file system is known in **accessingCluster**.

    gpfs Is the device name for the file system in **owningCluster**.

    owningCluster
    > Is the name of **owningCluster** as given by the **mmlscluster** command on a node in **owningCluster**.

    /mygpfs
    > Is the local mount point in **accessingCluster**.

11. On **accessingCluster**, the system administrator enters the **mmmount** command to mount the file system:

    `mmmount mygpfs`

Table 39 summarizes the commands that the administrators of the two clusters need to issue so that the nodes in **accessingCluster** can mount the remote file system **fs1**, which is owned by **owningCluster**, assigning **rfs1** as the local name with a mount point of **/rfs1**.

Table 39. Summary of commands to set up cross-cluster file system access.

accessingCluster	owningCluster
mmauth genkey new	**mmauth genkey new**
mmauth update . -l AUTHONLY	**mmauth update . -l AUTHONLY**
Exchange public keys (file **/var/mmfs/ssl/id_rsa.pub**)	Exchange public keys (file **/var/mmfs/ssl/id_rsa.pub**)
mmremotecluster add *owningCluster ...*	**mmauth add** *accessingCluster ...*
mmremotefs add rfs1 **-f** fs1 **-C** owningCluster **-T** /rfs1	**mmauth grant** *accessingCluster* **-f** fs1 ...

Managing remote access to a GPFS file system

This is an example of how to manage remote access to GPFS file systems.

To see a list of all clusters authorized to mount file systems owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth show
```

To authorize a third cluster, say **cluster3**, to access file systems owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth add cluster3 -k cluster3_id_rsa.pub
mmauth grant cluster3 -f /dev/gpfs1
```

To subsequently revoke **cluster3** authorization to access a specific file system **gpfs1** owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth deny cluster3 -f /dev/gpfs1
```

To completely revoke **cluster3** authorization to access file systems owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth delete cluster3
```

Using remote access with public and private IP addresses

GPFS permits the use of both public and private IP address. Private IP addresses are typically used to communicate on private networks.

Private IP addresses are on one of these subnets:
* 10.0.0.0
* 172.16.0.0
* 192.168.0.0

See RFC 1597 - Address Allocation for Private Internets(www.ip-doc.com/rfc/rfc1597) for more information.

Use the **mmchconfig** command, **subnets** attribute, to specify the private IP addresses to be accessed by GPFS.

Figure 12 on page 350 describes an AIX cluster named **CL1** with nodes named **CL1N1**, **CL1N2**, and so forth, a Linux cluster named **CL2** with nodes named **CL2N1**, **CL2N2**, and another Linux cluster named **CL3** with a node named **CL3N1**. Both Linux clusters have public Ethernet connectivity, and a Gigabit

Ethernet configured with private IP addresses (10.200.0.1 through 10.200.0.24), not connected to the public Ethernet. The InfiniBand Switch on the AIX cluster **CL1** is configured using public IP addresses on the 7.2.24/13 subnet and is accessible from the outside.

With the use of both public and private IP addresses for some of the nodes, the setup works as follows:

1. All clusters must be created using host names or IP addresses that correspond to the public network.

2. Using the **mmchconfig** command for the **CL1** cluster, add the attribute: **subnets=7.2.24.0**.

   This allows all **CL1** nodes to communicate using the InfiniBand Switch. Remote mounts between **CL2** and **CL1** will use the public Ethernet for TCP/IP communication, since the **CL2** nodes are not on the 7.2.24.0 subnet.

   GPFS assumes subnet specifications for private networks are independent between clusters (private networks are assumed not physically connected between clusters). The remaining steps show how to indicate that a private network is shared between clusters.

3. Using the **mmchconfig** command for the **CL2** cluster, add the **subnets='10.200.0.0/ CL2.kgn.ibm.com;CL3.kgn.ibm.com'** attribute. Alternatively, regular expressions are allowed here, such as **subnets='10.200.0.0/CL[23].kgn.ibm.com'**. See note 2 for the syntax allowed for the regular expressions.

   This attribute indicates that the private 10.200.0.0 network extends to all nodes in clusters **CL2** or **CL3**. This way, any two nodes in the **CL2** and **CL3** clusters can communicate through the Gigabit Ethernet.

   This setting allows all **CL2** nodes to communicate over their Gigabit Ethernet. Matching **CL3.kgn.ibm.com** with the cluster list for 10.200.0.0 allows remote mounts between clusters **CL2** and **CL3** to communicate over their Gigabit Ethernet.

4. Using the **mmchconfig** command for the **CL3** cluster, add the **subnets='10.200.0.0/ CL3.kgn.ibm.com;CL2.kgn.ibm.com'** attribute, alternatively **subnets='10.200.0.0/CL[32].kgn.ibm.com'**.

   This attribute indicates that the private 10.200.0.0 network extends to all nodes in clusters **CL2** or **CL3**. This way, any two nodes in the **CL2** and **CL3** clusters can communicate through the Gigabit Ethernet.

   Matching of **CL3.kgn.ibm.com** with the cluster list for 10.200.0.0 allows all **CL3** nodes to communicate over their Gigabit Ethernet, and matching **CL2.kgn.ibm.com** with that list allows remote mounts between clusters **CL3** and **CL2** to communicate over their Gigabit Ethernet.

Use the **subnets** attribute of the **mmchconfig** command when you wish the GPFS cluster to leverage additional, higher performance network connections that are available to the nodes in the cluster, or between clusters.

Notes:

1. Use of the **subnets** attribute does not ensure a highly available system. If the GPFS daemon is using the IP address specified by the **subnets** attribute, and that interface goes down, GPFS does not switch to the other network. You can use **mmdiag --network** to verify that the subnet is in fact being used.

2. Each subnet can be listed at most once in each cluster. For example, specifying:

   ```
   subnets='10.200.0.0/CL2.kgn.ibm.com 10.200.0.0/CL3.kgn.ibm.com'
   ```

   where the 10.200.0.0 subnet is listed twice, is not allowed. Therefore, subnets that span multiple clusters have to be assigned a cluster name pattern or a semicolon-separated cluster name list. It is possible to combine these, for example, items in semicolon-separated cluster lists can be plain names or regular expressions, as in the following:

   ```
   subnets='1.0.0.1/CL[23].kgn.ibm.com;OC.xyz.ibm.com'
   ```

   The following shows examples of patterns that are accepted:

   ```
   [af3] matches letters 'a' and 'f', and number 3
   [0-7] matches numbers 0, 1, ... 7
   [a-p0-7] matches letter a, b, ... p and numbers from 0 to 7 inclusive
   * matches any sequence of characters
   ? matches any (one) character
   ```

If the **subnets** attribute lists multiple subnets, and there are multiple subnets in common between the local cluster and a given remote cluster, then the first subnet in common in the list is used for communications between the local and remote clusters. As an example, suppose that the **subnets** attribute is set as follows, on cluster **CL2.kgn.ibm.com**:

```
subnets='10.200.0.0/CL[23].kgn.ibm.com 10.201.0.0/CL[23].kgn.ibm.com'
```

If node **CL2N1** on cluster **CL2.kgn.ibm.com** has network interfaces with IP addresses 10.200.0.1 and 10.201.0.1, and node **CLN31** on cluster **CL3.kgn.ibm.com** has network interfaces with IP addresses 10.200.0.5 and 10.201.0.5, then the communication between these two nodes will flow over the 10.200.0.0 subnet, with **CL2N1** using the interface with IP address 10.200.0.1, and **CLN31** using the interface with IP address 10.200.0.5.

Specifying a cluster name or a cluster name pattern for each subnet is only needed when a private network is shared across clusters. If the use of a private network is confined within the local cluster, then no cluster name is required in the subnet specification.
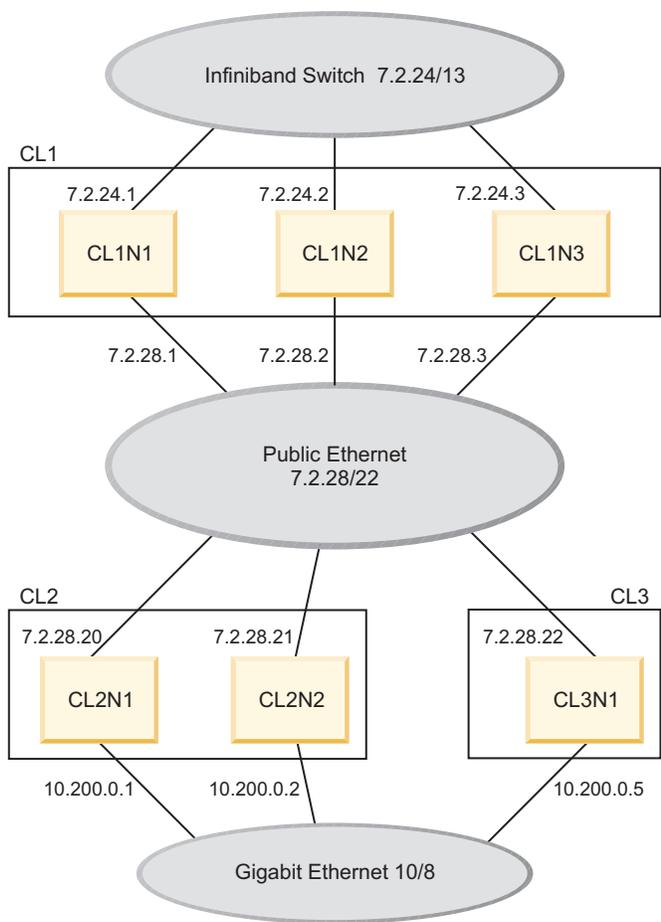

Figure 12. Use of public and private IP addresses in three GPFS clusters

Using multiple security levels for remote access

A cluster that owns a file system whose access is to be permitted from other clusters, can designate a different security level for each connecting cluster.

When multiple security levels are specified, each connection must use the security level of the connecting node unless that security level is **AUTHONLY**. In this case, the security level of the node that accepts the

connection is used instead. This means that a connection must use **AUTHONLY** if both nodes exist in clusters that are required to use the **AUTHONLY** security method.

To specify a different security level for different clusters that request access to a specified cluster, use the **mmauth -l** *cipherList* command. Several examples follow to illustrate:

1. In this example, **cluster1** and **cluster2** are on the same trusted network, and **cluster3** is connected to both of them with an untrusted network. The system administrator chooses these security levels:

   - A *cipherList* of **AUTHONLY** for connections between **cluster1** and **cluster2**
   - A *cipherList* of **AES128-SHA** for connections between **cluster1** and **cluster3**
   - A *cipherList* of **AES128-SHA** for connections between **cluster2** and **cluster3**

   The administrator of **cluster1** issues these commands:

   ```
   mmauth add cluster2 -k keyFile -l AUTHONLY
   mmauth add cluster3 -k keyFile -l AES128-SHA
   ```

2. In this example, **cluster2** is accessing file systems that are owned by **cluster1** by using a *cipherList* of **AUTHONLY**, but the administrator of **cluster1** decides to require a more secure *cipherList*. The administrator of **cluster1** issues this command:

   ```
   mmauth update cluster2 -l AES128-SHA
   ```

   Existing connections is upgraded from **AUTHONLY** to **AES128-SHA**.

Changing security keys with remote access

When working with GPFS file systems accessed by other GPFS clusters, it might be necessary to generate a new public/private access key. This can be done without disturbing existing connections, provided the following procedure is followed.

To accomplish this, the cluster that owns and serves the file system is made to temporarily have two access keys (referred to as the 'old key' and the 'new key'), which are both valid at the same time. The clusters currently accessing the file system can then change from the old key to the new key without interruption of file system access.

In this example, **cluster1** is the name of the cluster that owns and serves a file system, and **cluster2** is the name of the cluster that has already obtained access to this file system, and is currently using it. Here, the system administrator of **cluster1** changes the access key without severing the connection obtained by **cluster2**.

1. On **cluster1**, the system administrator issues the **mmauth genkey new** command to generate a new public/private access key pair. The key pair is placed in **/var/mmfs/ssl**:

   ```
   mmauth genkey new
   ```

   After this command is issued, **cluster1** will have two keys (referred to as the 'old key' and the 'new key') that can both be used to access **cluster1** file systems.

2. The system administrator of **cluster1** now gives the file **/var/mmfs/ssl/id_rsa.pub** (that contains the new key) to the system administrator of **cluster2**, who desires to continue to access the **cluster1** file systems. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.

3. On **cluster2**, the system administrator issues the **mmremotecluster update** command to make the new key known to his system:

   ```
   mmremotecluster update cluster1 -k cluster1_id_rsa.pub
   ```

   where:

   cluster1
   > Is the real name of **cluster1** as given by the **mmlscluster** command on a node in **cluster1**.

cluster1_id_rsa.pub

> Is the name of the file obtained from the administrator of **cluster1** in Step 2 on page 351.

This permits the cluster desiring to mount the file system to continue mounting file systems owned by **cluster1**.

4. On **cluster1**, the system administrator verifies that all clusters desiring to access **cluster1** file systems have received the new key and activated it using the **mmremotecluster update** command.

5. On **cluster1**, the system administrator issues the **mmauth genkey commit** command to commit the new key as the only valid access key. The old key will no longer be accepted once this command completes successfully:

```
mmauth genkey commit
```

Once the new public key has been committed, the old public key will no longer be accepted. As a result, any remote cluster administrator who has not been given the new key (see the preceding Step 2 on page 351) and run **mmremotecluster update** (see the preceding Step 3 on page 351) will no longer be able to mount file systems owned by **cluster1**.

Similarly, the administrator of **cluster2** might decide to change the access key for **cluster2**:

1. On **cluster2**, the system administrator issues the **mmauth genkey new** command to generate a new public/private access key pair. The key pair is placed in **/var/mmfs/ssl**:

```
mmauth genkey new
```

After this command is issued, **cluster2** will have two keys (referred to as the 'old key' and the 'new key') that can both be used when a connection is established to any of the nodes in **cluster2**.

2. The system administrator of **cluster2** now gives the file **/var/mmfs/ssl/id_rsa.pub** (that contains the new key) to the system administrator of **cluster1**, the owner of the file systems. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.

3. On **cluster1**, the system administrator issues the **mmauth update** command to make the new key known to his system:

```
mmauth update cluster2 -k cluster2_id_rsa.pub
```

where:

cluster2

> Is the real name of **cluster2** as given by the **mmlscluster** command on a node in **cluster2**.

cluster2_id_rsa.pub

> Is the name of the file obtained from the administrator of **cluster2** in Step 2.

This permits the cluster desiring to mount the file system to continue mounting file systems owned by **cluster1**.

4. The system administrator of **cluster2** verifies that the administrator of **cluster1** has received the new key and activated it using the **mmauth update** command.

5. On **cluster2**, the system administrator issues the **mmauth genkey commit** command to commit the new key as the only valid access key. The old key will no longer be accepted once this command completes successfully:

```
mmauth genkey commit
```

NIST compliance

The **nistCompliance** configuration variable allows the system administrator to restrict the set of available algorithms and key lengths to a subset of those approved by NIST.

About this task

The **nistCompliance** variable applies to security transport (tscomm security, key retrieval) only, not to encryption, which always uses NIST-compliant mechanisms.

For the valid values for **nistCompliance**, see *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference* guide.

The **nistCompliance** configuration variable has been introduced on version 4.1. Clusters created prior to that release operate with the equivalent of that variable being set to **off**. Similarly, clusters created on prior versions and which are migrated to 4.1 will have **nistCompliance** set to **off**.

Remote Mounts and version 3.5 clusters

A cluster created on version 4.1 or higher, and operating with **nistCompliance** set to **SP800-131A**, will be unable to remote-mount a file system from a version 3.5 cluster, since the 4.1 cluster will not accept the key from the latter, which is not NIST SP800-131A-compliant. To allow the version 4.1 cluster to remote-mount the version 3.5 cluster, issue the

```
mmchconfig nistCompliance=off
```

command on the version 4.1 cluster, before the **mmremotecluster add** command can be issued. The key exchange will work even if the version 4.1 cluster already has a NIST-compliant key.

Updating a cluster to nistCompliance SP800-131A

A cluster upgraded from prior versions may have the **nistCompliance** set to **off** and may be operating with keys which are not NIST SP800-131A-compliant. To upgrade the cluster to operate in NIST SP800-131A mode, the following procedure should be followed:

From a node in the cluster which is running version 4.1 or later, issue:

```
mmauth genkey new
mmauth genkey commit
```

If remote clusters are present, follow the procedure described in the "Changing security keys with remote access" on page 351 section (under Chapter 24, "Accessing a remote GPFS file system," on page 339) to update the key on the remote clusters.

Once all nodes in the cluster are running at least version 4.1, run the following command from one of the nodes in the cluster:

```
mmchconfig release=LATEST
```

From one of the nodes in the cluster, run the following command:

```
mmchconfig nistCompliance=SP800-131A
```

Important information about remote access

There is some additional information about this topic that you should take into consideration.

When working with GPFS file systems accessed by nodes that belong to other GPFS clusters, consider the following points:
1. A file system is administered only by the cluster where the file system was created. Other clusters may be allowed to mount the file system, but their administrators cannot add or delete disks, change characteristics of the file system, enable or disable quotas, run the **mmfsck** command, and so forth. The only commands that other clusters can issue are list type commands, such as: **mmlsfs**, **mmlsdisk**, **mmlsmount**, and **mmdf**.

2. Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters, like there is between the nodes within a cluster.

   This means that if the administrator of **cluster1** (the owner of file system **gpfs1**) decides to delete it or rename it, the information for **gpfs1** in **cluster2** becomes obsolete, and an attempt to mount **gpfs1** from **cluster2** will fail. It is assumed that when such changes take place, the two administrators will inform each other. The administrator of **cluster2** can then use the **update** or **delete** options of the **mmremotefs** command to make the appropriate changes.

3. If the names of the contact nodes change, the name of the cluster changes, or the public key file changes, use the **update** option of the **mmremotecluster** command to reflect the changes.

4. Use the **show** option of the **mmremotecluster** and **mmremotefs** commands to display the current information about remote clusters and file systems.

5. If the cluster that owns a file system has a **maxblocksize** configuration parameter that is different from the **maxblocksize** configuration parameter of the cluster that desires to mount a file system, a mismatch may occur and file system mount requests may fail with messages to this effect. Check your **maxblocksize** configuration parameters on both clusters using the **mmlsconfig** command. Correct any discrepancies with the **mmchconfig** command.

Chapter 25. Information lifecycle management for IBM Spectrum Scale

IBM Spectrum Scale can help you achieve information lifecycle management (ILM) efficiencies through powerful policy-driven automated tiered storage management. With the ILM toolkit, you can manage sets of files and pools of storage, and you can automate the management of file data.

Using these tools, GPFS can automatically determine where to physically store your data regardless of its placement in the logical directory structure. Storage pools, filesets and user-defined policies provide the ability to match the cost of your storage resources to the value of your data.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

GPFS policy-based ILM tools allow you to:
- Create *storage pools* to provide a way to partition a file system's storage into collections of disks or a redundant array of independent disks (RAIDs) with similar properties that are managed together as a group. GPFS has three types of storage pools:
  - A required **system** storage pool that you create and manage through GPFS
  - Optional user storage pools that you create and manage through GPFS
  - Optional external storage pools that you define with GPFS policy rules and manage through an external application such as IBM Spectrum Protect
- Create *filesets* to provide a way to partition the file system namespace to allow administrative operations at a finer granularity than that of the entire file system. See "Filesets" on page 402.
- Create *policy rules* based on data attributes to determine initial file data placement and manage file data placement throughout the life of the file. See "Policies for automating file management" on page 362.

To work with ILM in the GUI, click **Files** > **Information Lifecycle**.

Use the following information to create and manage information lifecycle management policies in IBM Spectrum Scale:

Storage pools

Physically, a *storage pool* is a collection of disks or RAID arrays. Storage pools also allow you to group multiple storage systems within a file system.

Using storage pools, you can create tiers of storage by grouping storage devices based on performance, locality, or reliability characteristics. For example, one pool could be an enterprise class storage system that hosts high-performance Fibre Channel disks and another pool might consist of numerous disk controllers that host a large set of economical SATA disks.

There are two types of storage pools in GPFS, internal storage pools and external storage pools. Internal storage pools are managed within GPFS. External storage pools are managed by an external application such as IBM Spectrum Protect. For external storage pools, GPFS provides tools that allow you to define an interface that your external storage manager uses to access your data. GPFS does not manage the data placed in external storage pools. Instead, GPFS manages the movement of data to and from external storage pools. Storage pools allow you to perform complex operations such as moving, mirroring, or deleting files across multiple storage devices, providing storage virtualization and a single management context.

Internal GPFS storage pools are meant for managing online storage resources. External storage pools are intended for use as near-line storage and for archival and backup operations. However, both types of storage pools provide you with a method to partition file system storage for considerations such as:

* Improved price-performance by matching the cost of storage to the value of the data
* Improved performance by:
  – Reducing the contention for premium storage
  – Reducing the impact of slower devices
  – Allowing you to retrieve archived data when needed
* Improved reliability by providing for:
  – Replication based on need
  – Better failure containment
  – Creation of new storage pools as needed

For additional information, refer to:
* "Internal storage pools"
* "External storage pools" on page 361

Internal storage pools

Internal GPFS storage pools are controlled by GPFS policies and commands. There are two types of internal GPFS storage pools, the required system storage pool and up to seven optional user storage pools. The system storage pool contains metadata for each file and may also contain user data. User storage pools can only contain user data.

The internal GPFS storage pool to which a disk belongs is specified as an attribute of the disk in the GPFS cluster. You specify the disk attributes as a field in each disk descriptor when you create the file system or when adding disks to an existing file system. GPFS allows a maximum of eight internal storage pools per file system. One of these storage pools is the required **system** storage pool. The other seven internal storage pools are optional user storage pools.

GPFS assigns file data to internal storage pools under these circumstances:

* When the file is initially created; the storage pool is determined by the file placement policy that is in effect when at the time of file creation.
* When the attributes of the file, such as file size or access time, match the rules of a policy that directs GPFS to migrate the data to a different storage pool.

For additional information, refer to:
* "The system storage pool"
* "The system.log storage pool" on page 357
* "User storage pools" on page 357
* "Managing storage pools" on page 358

The system storage pool

The **system** storage pool contains file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks, extended attributes, and so forth.

The **system** storage pool can also contain user data. There is only one **system** storage pool per file system, and it is automatically created when the file system is created.

Important: It is recommended that you use highly-reliable disks and replication for the **system** storage pool because it contains system metadata.

The amount of metadata grows as you add files to the system. Therefore, it is recommended that you monitor the **system** storage pool to ensure that there is always enough space to accommodate growth. The **system** storage pool typically requires a small percentage of the total storage capacity that GPFS manages. However, the percentage required by the **system** storage pool varies depending on your environment. You can monitor the amount of space available in the **system** storage pool with the **mmdf** command. If the available space in the system storage pool begins to run low, you can increase the available space by purging files or adding disks to the system storage pool.

The system.log storage pool

By default the file system recovery log is stored in the system storage pool with file system metadata. The file system recovery log can also be placed in a dedicated pool that is called the **system.log** pool.

This storage pool must be created explicitly. It is highly recommended to only use storage that is as fast or even faster than what is used for the system storage pool. This recommendation is because of the high number of small synchronous data updates made to the recovery log. The block size for the **system.log** pool must be the same as the block size of the system pool.

The file system recovery log will only be stored in one pool.

The system.log storage pool is an optional dedicated storage pool that contains only the file system recovery logs. If you define this pool, then IBM Spectrum Scale uses it for all the file system recovery logs of the file system. Otherwise, the file system recovery logs are kept in the system storage pool. It is a good practice for the system.log pool to consist of storage media that is as fast as or faster than the storage media of the system storage pool. If the storage is nonvolatile, this pool can be used for the high-availability write cache (HAWC).

User storage pools

All user data for a file is stored in the assigned storage pool as determined by your file placement rules.

In addition, file data can be migrated to a different storage pool according to your file management policies. For more information on policies, see "Policies for automating file management" on page 362.

A user storage pool *only* contains the blocks of data (user data, for example) that make up a user file. GPFS stores the data that describes the files, called *file metadata*, separately from the actual file data in the **system** storage pool. You can create one or more user storage pools, and then create policy rules to indicate where the data blocks for a file should be stored.

Tracking file access temperature within a storage pool

A file's access temperature is an attribute for policy that provides a means of optimizing tiered storage. File temperatures are a relative attribute, indicating whether a file is "hotter" or "colder" than the others in its pool. The policy can be used to migrate hotter files to higher tiers and colder files to lower. The access temperature is an exponential moving average of the accesses to the file. As files are accessed the temperature increases; likewise when the access stops the file cools. File temperature is intended to optimize nonvolatile storage, not memory usage; therefore, cache hits are not counted. In a similar manner, only user accesses are counted.

The access counts to a file are tracked as an exponential moving average. An unaccessed file loses a percentage of its accesses each period. The loss percentage and period are set via the configuration variables **fileHeatLossPercent** and **fileHeatPeriodMinutes**. By default, the file access temperature is not tracked. To use access temperature in policy, the tracking must first be enabled. To do this, set the two configuration variables as follows:

fileHeatLossPercent
> The percentage (between 0 and 100) of file access temperature dissipated over the **fileHeatPeriodMinutes** time. The default value is 10.

fileHeatPeriodMinutes
> The number of minutes defined for the recalculation of file access temperature. To turn on tracking, **fileHeatPeriodMinutes** must be set to a nonzero value. The default value is 0.

The following example sets **fileHeatPeriodMinutes** to 1440 (24 hours) and **fileHeatLossPercent** to 10, meaning that unaccessed files will lose 10% of their heat value every 24 hours, or approximately 0.4% every hour (because the loss is continuous and "compounded" geometrically):

```
mmchconfig fileheatperiodminutes=1440,fileheatlosspercent=10
```

Note: If the updating of the file access time (atime) is suppressed or if relative atime semantics are in effect, proper calculation of the file access temperature may be adversely affected.

File access temperature is tracked on a per-cluster basis, not on a per–file system basis.

Use **WEIGHT(FILE_HEAT)** with a policy **MIGRATE** rule to prioritize migration by file temperature. (You can use the **GROUP POOL** rule to define a group pool to be specified as the **TO POOL** target.) See "Policies for automating file management" on page 362.

Managing storage pools

Managing your storage pools includes:
- "Creating storage pools"
- "Changing the storage pool assignment of a disk" on page 359
- "Changing the storage pool assignment of a file" on page 359
- "Deleting storage pools" on page 359
- "Listing the storage pools of a file system" on page 359
- "Listing the storage pool of a file" on page 360
- "Listing disks and associated statistics" on page 360
- "Rebalancing files in a storage pool" on page 361
- "Using replication in a storage pool" on page 361

Creating storage pools:

The storage pool to which a disk belongs is an attribute of each disk and is specified as a field in each disk descriptor when the file system is created using the **mmcrfs** command or when disks are added to an existing file system with the **mmadddisk** command. Adding a disk with a new storage pool name in the disk descriptor automatically creates the storage pool.

Storage pool names:
- Must be unique within a file system, but not across file systems.
- Cannot be longer than 255 alphanumeric characters.
- Are case sensitive. **MYpool** and **myPool** are distinct storage pools.

A storage pool is defined by the stanza keyword **pool**; for example:
```
pool=dataPoolA
```

If a storage pool is not specified, the disk is by default assigned to the **system** storage pool.

The **--metadata-block-size** flag on the **mmcrfs** command can be used to create a system pool with a different block size from the user pools. This can be especially beneficial if the default block size is larger than 1 MB. If data and metadata block sizes differ, the system pool must contain only **metadataOnly** disks.

Changing the storage pool assignment of a disk:

Once a disk is assigned to a storage pool, the pool assignment cannot be changed by using either the **mmchdisk** command or the **mmrpldisk** command. You can, however, change the pool to which the disk is assigned.

To move a disk to another pool:
1. Delete the disk from its current pool by issuing the **mmdeldisk** command. This will move the data to the remaining disks in the storage pool.
2. Add the disk to the new pool by issuing the **mmadddisk** command.
3. Rebalance the data across all disks in the new storage pool by issuing the **mmrestripefs -P** command.

Changing the storage pool assignment of a file:

You can change the storage pool that a file is assigned to.

A root user can change the storage pool that a file is assigned to by either:
* Running **mmapplypolicy** with an appropriate set of policy rules.
* Issuing the **mmchattr -P** command.

By default, both of these commands migrate data immediately (this is the same as using the **-I yes** option for these commands). If desired, you can delay migrating the data by specifying the **-I defer** option for either command. Using the defer option, the existing data does not get moved to the new storage pool until either the **mmrestripefs** command or the **mmrestripefile** command are executed. For additional information, refer to:
* "Overview of policies" on page 362
* "Rebalancing files in a storage pool" on page 361

Deleting storage pools:

System storage pools, **system.log** pools and user storage pools have different deletion requirements.

Deleting the **System** storage pool is not allowed. You can delete the **system** storage pool only after you have deleted the file system.

You can delete the **system.log** pool by deleting all the disks in the **system.log** pool. You do not need to run a policy to empty the **system.log** pool first, because the **system.log** pool can only contain log files, and those are automatically migrated to the **System** pool when you delete the **System** pool.

In order to delete a user storage pool, you must delete all its disks using the **mmdeldisk** command. When GPFS deletes the last remaining disk from a user storage pool, the storage pool is also deleted. To delete a storage pool, it must be completely empty. A migration policy along with the **mmapplypolicy** command could be used to do this.

Listing the storage pools of a file system:

To list the storage pools available for a specific file system, issue the **mmlsfs -P** command.

For example, this command:
```
mmlsfs fs1 -P
```

produces output similar to this:

```
flag               value                   description
------------------ ----------------------- ------------------------------------
 -P                system;sp1;sp2          Disk storage pools in file system
```

For file system **fs1**, there are three storage pools: the **system** storage pool and user storage pools named **sp1** and **sp2**.

Listing the storage pool of a file:

To display the assigned storage pool and the name of the fileset that includes the file, issue the **mmlsattr -L** command.

For example, this command:
```
mmlsattr -L myfile
```

produces output similar to this:
```
file name:            myfile
metadata replication: 2 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    sp1
fileset name:         root
snapshot name:
creation Time:        Wed Feb 22 15:16:29 2012
Misc attributes:      ARCHIVE
```

File **myfile** is assigned to the storage pool named **sp1** and is part of the root fileset.

Listing disks and associated statistics:

To list the disks belonging to a storage pool, issue the **mmdf -P** command.

For example, this command:
```
mmdf fs1 -P sp1
```

produces output similar to this:
```
disk            disk size failure holds    holds              free KB              free KB
name               in KB    group metadata data      in full blocks         in fragments
--------------- ---------- ------- -------- ----- -------------------- --------------------
Disks in storage pool: sp1 (Maximum disk size allowed is 1.2 TB)
vp4vsdn05         17760256       6 no       yes        11310080 ( 64%)         205200 ( 1%)
vp5vsdn05         17760256       6 no       yes        11311104 ( 64%)         205136 ( 1%)
vp6vsdn05         17760256       6 no       yes        11300352 ( 64%)         206816 ( 1%)
vp7vsdn05         17760256       6 no       yes        11296256 ( 64%)         209872 ( 1%)
vp0vsdn05         17760256       6 no       yes        11293696 ( 64%)         207968 ( 1%)
vp1vsdn05         17760256       6 no       yes        11293184 ( 64%)         206464 ( 1%)
vp2vsdn05         17760256       6 no       yes        11309056 ( 64%)         203248 ( 1%)
vp3vsdn05         17760256       6 no       yes        11269120 ( 63%)         211456 ( 1%)
                  -------------                        -------------------- --------------------
(pool total)     142082048                             90382848 ( 64%)       1656160 ( 1%)
```

This example shows that storage pool **sp1** in file system **fs1** consists of eight disks and identifies details for each disk including:
- Name
- Size
- Failure group
- Data type

- Free space

Rebalancing files in a storage pool:

A root user can rebalance file data across all disks in a file system by issuing the **mmrestripefs** command.

Optionally:
- Specifying the **-P** option rebalances only those files assigned to the specified storage pool.
- Specifying the **-p** option rebalances the file placement within the storage pool. For files that are assigned to one storage pool, but that have data in a different pool, (referred to as ill-placed files), using this option migrates their data to the correct pool. (A file becomes "ill-placed" when the **-I defer** option is used during migration of the file between pools.)

Using replication in a storage pool:

To enable data replication in a storage pool, you must make certain that there are at least two failure groups within the storage pool.

This is necessary because GPFS maintains separation between storage pools and performs file replication within each storage pool. In other words, a file and its replica must be in the same storage pool. This also means that if you are going to replicate the entire file system, every storage pool in the file system must have at least two failure groups.

Note: Depending on the configuration of your file system, if you try to enable file replication in a storage pool having only one failure group, GPFS will either give you a warning or an error message.

External storage pools

When you initially create a file, GPFS assigns that file to an internal storage pool. Internal storage pools support various types of online storage. To move data from online storage to offline or near-line storage, you can use external storage pools.

External storage pools use a flexible interface driven by GPFS policy rules that simplify data migration to and from other types of storage such as tape storage. For additional information, refer to "Policies for automating file management" on page 362.

You can define multiple external storage pools at any time using GPFS policy rules. To move data to an external storage pool, the GPFS policy engine evaluates the rules that determine which files qualify for transfer to the external pool. From that information, GPFS provides a list of candidate files and executes the script specified in the rule that defines the external pool. That executable script is the interface to the external application, such as IBM Spectrum Protect, that does the actual migration of data into an external pool. Using the external pool interface, GPFS gives you the ability to manage information by allowing you to:
1. Move files and their extended attributes onto low-cost near-line or offline storage when demand for the files diminishes.
2. Recall the files, with all of their previous access information, onto online storage whenever the files are needed.

External pool requirements

With external pools, GPFS provides metadata processing and the flexibility of using extended file attributes. The external storage manager is responsible for moving files from GPFS and returning them upon the request of an application accessing the file system. Therefore, when you are using external storage pools, you must use an external file management application such as IBM Spectrum Protect. The external application is responsible for maintaining the file once it has left the GPFS file system. For example, GPFS policy rules create a list of files that are eligible for migration. GPFS hands that list to

IBM Spectrum Protect which migrates the files to tape and creates a reference file in the file system that has pointers to the tape image. When a file is requested, it is automatically retrieved from the external storage pool and placed back in an internal storage pool. As an alternative, you can use a GPFS policy rule to retrieve the data in advance of a user request.

The number of external storage pools is only limited by the capabilities of your external application. GPFS allows you to define external storage pools at any time by writing a policy that defines the pool and makes that location known to GPFS. External storage pools are defined by policy rules and initiated by either storage thresholds or use of the **mmapplypolicy** command.

For additional information, refer to "Working with external storage pools" on page 394.

Policies for automating file management

GPFS provides a means to automate the management of files using policies and rules. Properly managing your files allows you to efficiently use and balance your premium and less expensive storage resources.

GPFS supports these policies:
- *File placement policies* are used to automatically place newly created files in a specific storage pool.
- Snapshot placement policies are used to automatically place snapshot data in a specific storage pool.
- *File management policies* are used to manage files during their lifecycle by moving them to another storage pool, moving them to near-line storage, copying them to archival storage, changing their replication status, or deleting them. They can also be used to migrate snapshot data to other storage pools or change its replication status.
- *Transparent Cloud Tiering policies* are used to migrate cold data to a cloud storage tier or recall data from the cloud storage tier on reaching certain threshold levels.

You can create and manage policies and policy rules with both the command line interface and the GUI. In the GUI, navigate to **Files** > **Information Lifecycle Management**.

Overview of policies

A *policy* is a set of rules that describes the life cycle of user data based on the attributes of files. Each rule defines an operation or definition, such as "migrate to a pool and replicate the file."

You can perform the following tasks with rules:
- Initial file placement
- Snapshot data placement
- File management
- Restoring file data
- Encryption-specific uses. For more information, see the topic *Encryption* in the *IBM Spectrum Scale: Command and Programming Reference*.
- File compression and decompression. For more information, see "File compression" on page 119.

When a file is created or restored, the placement policy determines the location of the file's data and assigns the file to a storage pool. All data written to that file is placed in the assigned storage pool.

Similarly, if the file system has snapshots and a file is written to, the snapshot placement policy determines the storage pool where the snapshot blocks are placed.

The placement policy defining the initial placement of newly created files, snapshot data, and the rules for placement of restored data must be installed into GPFS with the **mmchpolicy** command. If a GPFS file system does not have a placement policy installed, all the data is stored in the first data storage pool. Only one placement policy can be installed at a time. If you switch from one placement policy to another,

or make changes to a placement policy, that action has no effect on existing files. However, newly created files are always placed according to the currently installed placement policy.

The management policy determines file management operations such as migration, deletion, and file compression or decompression.

In order to migrate or delete data, you must use the **mmapplypolicy** command. To compress or decompress data, you can use either the **mmapplypolicy** command with a **MIGRATE** rule or the **mmchattr** command. You can define the file management rules and install them in the file system together with the placement rules. As an alternative, you may define these rules in a separate file and explicitly provide them to **mmapplypolicy** using the **-P** option. In either case, policy rules for placement or migration may be intermixed. Over the life of the file, data can be migrated to a different storage pool any number of times, and files can be deleted or restored.

Note: In a multicluster environment, the scope of the **mmapplypolicy** command is limited to the nodes in the cluster that owns the file system.

Note: File compression or decompression using the **mmapplypolicy** command is not supported on the Windows operating system.

File management rules can also be used to control the space utilization of GPFS online storage pools. When the utilization for an online pool exceeds the specified high threshold value, GPFS can be configured, through user exits, to trigger an event that can automatically start **mmapplypolicy** and reduce the utilization of the pool. Using the **mmaddcallback** command, you can specify a script that will run when such an event occurs. For more information, see the topic *mmaddcallback command* in the *IBM Spectrum Scale: Command and Programming Reference*.

GPFS performs error checking for file-placement policies in the following phases:
- When you install a new policy, GPFS checks the basic syntax of all the rules in the policy.
- GPFS also checks all references to storage pools. If a rule in the policy refers to a storage pool that does not exist, the policy is not installed and an error is returned.
- When a new file is created, the rules in the active policy are evaluated in order. If an error is detected, GPFS logs an error, skips all subsequent rules, and returns an **EINVAL** error code to the application.
- Otherwise, the first applicable rule is used to store the file data.

Default file placement policy:
> When a GPFS file system is first created, the default file placement policy is to assign all files to the **system** storage pool. You can go back to the default policy by running the command:
> ```
> mmchpolicy Device DEFAULT
> ```

For more information on using GPFS commands to manage policies, see "Managing policies" on page 391.

Policy rules

A *policy rule* is an SQL-like statement that tells GPFS what to do with the data for a file in a specific storage pool if the file meets specific criteria. A rule can apply to any file being created or only to files being created within a specific fileset or group of filesets.

Rules specify conditions that, when true, cause the rule to be applied. Conditions that cause GPFS to apply a rule include:
- Date and time when the rule is evaluated, that is, the current date and time
- Date and time when the file was last accessed
- Date and time when the file was last modified
- Fileset name

* File name or extension
* File size
* User ID and group ID

GPFS evaluates policy rules in order, from first to last, as they appear in the policy. The first rule that matches determines what is to be done with that file. For example, when a client creates a file, GPFS scans the list of rules in the active file placement policy to determine which rule applies to the file. When a rule applies to the file, GPFS stops processing the rules and assigns the file to the appropriate storage pool. If no rule applies, an EINVAL error code is returned to the application.

There are nine types of policy rules that allow you to define specific actions that GPFS will implement on the file data. Each rule has clauses that control candidate selection, namely when the rule is allowed to match a file, what files it will match, the order to operate on the matching files and additional attributes to show for each candidate file. Different clauses are permitted on different rules based upon the semantics of the rule.

Policy rules: Syntax

Policy rules can apply to file placements, snapshot placements, group pools, file migrations, file deletions, file exclusions, file lists, file restores, external storage pool definitions, and external list definitions.

The policy rules and their respective syntax diagrams are as follows. For more information about encryption-specific rules, see Chapter 35, "Encryption," on page 557.

* File placement rules

```
RULE ['RuleName']
  SET POOL 'PoolName'
    [LIMIT (OccupancyPercentage)]
    [REPLICATE (DataReplication)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

* Snapshot placement rule

```
RULE ['RuleName']
SET SNAP_POOL 'PoolName'
[LIMIT (OccupancyPercentage)]
[REPLICATE (DataReplication)]
[FOR FILESET ('FilesetName'[,'FilesetName']...)]
[ACTION (SqlExpression)]
[WHERE SqlExpression]
```

* Group pool rule; used to define a list of pools that may be used as a pseudo-pool source or destination in either a **FROM POOL** or **TO POOL** clause within another rule

```
RULE ['RuleName'] GROUP POOL ['groupPoolName']
IS 'poolName' [LIMIT(OccupancyPercentage)]
THEN 'poolName2' [LIMIT(n2)]
THEN 'pool-C' [LIMIT(n3)]
THEN ...
```

* File migration rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  MIGRATE [COMPRESS ({'yes' | 'no' | 'lz4' | 'z'})]
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage[,PremigratePercentage]])]
    [WEIGHT (WeightExpression)]
  TO POOL 'ToPoolName'
    [LIMIT (OccupancyPercentage)]
    [REPLICATE (DataReplication)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

For more information, see the topic "File compression" on page 119.

- File deletion rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  DELETE
    [DIRECTORIES_PLUS]
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage])]
    [WEIGHT (WeightExpression)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

- File exclusion rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  EXCLUDE
   [DIRECTORIES_PLUS]
   [FROM POOL 'FromPoolName']
   [FOR FILESET ('FilesetName'[,'FilesetName']...)]
   [ACTION (SqlExpression)]
   [WHERE SqlExpression]
```

- File list rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  LIST 'ListName'
    [EXCLUDE]
    [DIRECTORIES_PLUS]
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage])]
    [WEIGHT (WeightExpression)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

- File restore rule

```
RULE ['RuleName']
  RESTORE TO POOL 'PoolName'
    [LIMIT (OccupancyPercentage)]
    [REPLICATE (DataReplication)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

- External storage pool definition rule

```
RULE ['RuleName']
  EXTERNAL POOL 'PoolName'
   EXEC 'InterfaceScript'
   [OPTS 'OptionsString ...']
   [ESCAPE '%SpecialCharacters']
   [SIZE sum-number]
```

- External list definition rule

```
RULE ['RuleName']
  EXTERNAL LIST 'ListName'
   EXEC 'InterfaceScript'
   [OPTS 'OptionsString ...']
   [ESCAPE '%SpecialCharacters']
   [THRESHOLD 'ResourceClass']
   [SIZE sum-number]
```

Policy rules: Terms
The terms of policy rules specify conditions for selecting files and operations to perform on files.

The following terms are used in policy rules. Some terms appear in more than one rule:

ACTION (*SqlExpression***)**
Specifies an SQL expression that is evaluated only if the other clauses of the rule are satisfied. The action of the *SqlExpression* is completed, and the resulting value of the *SqlExpression* is discarded. In the following example, the rule sets the extended attribute "user.action" to the value "set pool s6" for files that begin with the characters "sp". These files are assigned to the system pool:

```
rule 's6' set pool 'system' action(setxattr('user.action','set pool s6')) where name like 'sp%'
```

Note: Encryption policies do not support the **ACTION** clause.

[COMPRESS ({'yes' | 'no' | 'lz4' | 'z'})]
Indicates that the selected files are to be compressed or decompressed.

Note: Compression with the z compression library is intended primarily for cold data and favors saving space over access speed. Compression with the lz4 compression library is intended primarily for active data and favors access speed over saving space.

yes
Files that are uncompressed are to be compressed with the z compression library. Files that are already compressed are not affected.

no Files that are compressed are to be decompressed. Files that are already uncompressed are not affected.

z Files that are uncompressed are to be compressed with the z compression library. Files that are already compressed with the lz4 library are to be recompressed with the z library. Files that are already compressed with the z library are not affected.

lz4
Files that are uncompressed are to be compressed with the lz4 compression library. Files that are already compressed with the z library are to be recompressed with the lz4 library. Files that are already compressed with the lz4 library are not affected.

The following rule compresses the files in the pool datapool that begin with the string green%. Because the policy term **COMPRESS** specifies **yes** instead of a compression library, compression is done with the default compression library, which is the z library.

```
RULE 'COMPR1' MIGRATE FROM POOL 'datapool' COMPRESS('yes') WHERE NAME LIKE 'green%'
```

For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration Guide*.

DIRECTORIES_PLUS
Indicates that non-regular file objects (directories, symbolic links, and other items) must be included in the list. If not specified, only ordinary data files are included in the candidate lists.

DELETE
Identifies a file deletion rule. A file that matches this rule becomes a candidate for deletion.

ESCAPE '%*SpecialCharacters***'**
Specifies that path names and the **SHOW**('*string*') expressions within the associated file lists are encoded with a scheme based on RFC3986 URI percent encoding.

Compare the two following rules:
```
RULE 'xp' EXTERNAL POOL 'pool-name' EXEC 'script-name' ESCAPE '%'
RULE 'xl' EXTERNAL LIST 'list-name' EXEC 'script-name' ESCAPE '%/+@#'
```

Both rules specify that all characters except the "unreserved" characters in the set `a-zA-Z0-9-_.~` are encoded as `%XX`, where XX comprises two hexadecimal digits.

However, the GPFS **ESCAPE** syntax adds to the set of "unreserved" characters. In the first rule, the syntax ESCAPE '%' specifies a rigorous RFC3986 encoding. Under this rule, a path name such as `/root/directory/@abc+def#ghi.jkl` appears in a file list in the following format:

`%2Froot%2Fdirectory%2F%40abc%2Bdef%23ghi.jkl`

In the second rule, the syntax ESCAPE '%/+@#' specifies that none of the characters in set /+@# are escaped. Under this rule, the same path name appears in a file list in the following format:

`/root/directory/@abc+def#ghi.jkl`

If you omit the **ESCAPE** clause, the newline character is escaped as '\n', and the backslash character is escaped as '\\'; all other characters are presented as is, without further encoding.

EXCLUDE
Identifies a file exclusion rule.

> **RULE** `'x'` **EXCLUDE**
> A file that matches this form of the rule is excluded from further consideration by any **MIGRATE** or **DELETE** rules that follow.

> **RULE** `'rule-name'` **LIST** `'listname-y'` **EXCLUDE**
> A file that matches this form of the rule is excluded from further consideration by any **LIST** rules that name the same *listname-y*.

EXEC `'InterfaceScript'`
Specifies an external program to be invoked to pass requests to an external storage management application. *InterfaceScript* must be a fully qualified path name to a user-provided script or program that supports the commands described in "User-provided program for managing external pools" on page 395.

EXTERNAL LIST `ListName`
Defines an external list. This rule does not match files. It provides the binding between the lists that are generated with regular **LIST** rules with a matching *ListName* and the external program that you want to run with these lists as input.

EXTERNAL POOL `PoolName`
Defines an external storage pool. This rule does not match files but defines the binding between the policy language and the external storage manager that implements the external storage.

FOR FILESET (`'FilesetName'`**[,**`'FilesetName'`**]...)**
Specifies that the rule applies only to files within the specified filesets.

FROM POOL `FromPoolName`
Specifies the name of the source pool from which files are candidates for migration.

GROUP POOL `PoolName`
Defines a group pool. This rule supports the concept of distributing data files over several GPFS disk pools.

Optionally, a **LIMIT**, specified as an occupancy percentage, can be specified for each disk pool; if not specified, the limit defaults to 99%. The **THEN** keyword signifies that disk pools that are specified before a **THEN** keyword are preferred over disk pools that are specified after. When a pool that is defined by a **GROUP POOL** rule is the **TO POOL** target of a **MIGRATE** rule, the selected files are distributed among the disk pools that comprise the group pool. Files of highest weight are put into the most preferred disk pool up to the occupancy limit for that pool. If more files must be migrated, they are put into the second most preferred pool up to the occupancy limit for that pool. Again, files of highest weight are selected.

If you specify a file that is defined by a **GROUP POOL** rule in a **FROM POOL** clause, the clause matches any file in any of the disk pools in the group pool.

You can "repack" a group pool by **WEIGHT**. Migrate files of higher weight to preferred disk pools by specifying a group pool as both the source and the target of a **MIGRATE** rule.

```
rule 'grpdef' GROUP POOL 'gpool' IS 'ssd' LIMIT(90) THEN 'fast' LIMIT(85) THEN 'sata'
rule 'repack' MIGRATE FROM POOL 'gpool' TO POOL 'gpool' WEIGHT(FILE_HEAT)
```

See "Tracking file access temperature within a storage pool" on page 357.

LIMIT (*OccupancyPercentage***)**

Limits the creation of data in a storage pool. GPFS does not migrate a file into a pool if doing so exceeds the occupancy percentage for the pool. If you do not specify an occupancy percentage for a pool, the default value is 99%. See "Phase two: Choosing and scheduling files" on page 384.

You can specify *OccupancyPercentage* as a floating point number, as in the following example:

```
RULE 'r' RESTORE to pool 'x' limit(8.9e1)
```

For testing or planning purposes, and when you use the **mmapplypolicy** command with the **-I defer** or **-I test** option, you can specify a **LIMIT** larger than 100%.

The limit clause does not apply when the target **TO POOL** is a **GROUP POOL**. The limits that are specified in the rule that defines the target **GROUP POOL** govern the action of the **MIGRATE** rule.

LIST *ListName*

Identifies a file list generation rule. A file can match more than one list rule but appears in a list only once. *ListName* provides the binding to an **EXTERNAL LIST** rule that specifies the executable program to call when the generated list is processed.

MIGRATE

Identifies a file migration rule. A file that matches this rule becomes a candidate for migration to the pool specified by the **TO POOL** clause.

OPTS '*OptionsString ...***'**

Specifies optional parameters to be passed to the external program defined with the **EXEC** clause. *OptionsString* is not interpreted by the GPFS policy engine.

REPLICATE (*DataReplication***)**

Overrides the default data replication factor. This value must be specified as 1, 2, or 3.

RESTORE TO POOL *PoolName*

where Identifies a file restore rule. When you restore a file with the **gpfs_fputattrswithpathname()** subroutine, you can use this rule to match files against their saved attributes rather than the current file attributes. This rule also applies to a command that uses that subroutine, such as the IBM Spectrum Protect command **dsmc restore**.

RULE ['*RuleName***']**

Initiates the rule statement. *RuleName* identifies the rule and is used in diagnostic messages.

SET POOL {*PoolName* **| EXCLUDE}**

Identifies an initial file placement rule.

PoolName

Specifies the name of the storage pool where all files that match the rule criteria are placed.

EXCLUDE

Specifies that files that match the rule criteria are excluded from further consideration and will not be stored in any pool. If you try to create a file that matches a **SET POOL EXCLUDE** rule, the file system API returns the error ENOSPC.

SET SNAP_POOL *PoolName*

Identifies an initial snapshot placement rule. *PoolName* specifies the name of the storage pool where all snapshot files that match the rule criteria are placed.

Note: The pool is only set when the file data is written to the snapshot, not when the snapshot is created.

SHOW (['*String*'] *SqlExpression*)
Inserts the requested information (the character representation of the evaluated SQL expression *SqlExpression*) into the candidate list that is created by the rule when it deals with external storage pools. *String* is a literal value that gets echoed back.

This clause has no effect in matching files but can be used to define other attributes to be exported with the candidate file lists.

SIZE (*numeric-sql-expression*)
Is an optional clause of any **MIGRATE**, **DELETE**, or **LIST** rules that are used for choosing candidate files. *numeric-sql-expression* specifies the size of the file to be used when in calculating the total amount of data to be passed to a user script. The default is **KB_ALLOCATED**.

SIZE *sum-number*
Is an optional clause of the **EXTERNAL POOL** and **EXTERNAL LIST** rules. *sum-number* limits the total number of bytes in all of the files named in each list of files passed to your EXEC 'script'. If a single file is larger than *sum-number*, it is passed to your EXEC 'script' as the only entry in a "singleton" file list.

Specify *sum-number* as a numeric constant or a floating-point value.

Note: The value of *sum-number* is in kilobytes.

THRESHOLD (*HighPercentage*[,*LowPercentage*[,*PremigratePercentage*]])
Controls migration and deletion based on the percent of assigned pool storage that is occupied.

HighPercentage
Indicates that the rule is to be applied only if the occupancy percentage of the current pool of the file is greater than or equal to the *HighPercentage* value. Specify a nonnegative integer in the range 0-100.

LowPercentage
Indicates that **MIGRATE** and **DELETE** rules are to be applied until the occupancy percentage of the current pool of the file is reduced to less than or equal to the *LowPercentage* value. Specify a nonnegative integer in the range 0-100. The default is 0%.

PremigratePercentage
Defines an occupancy percentage of a storage pool that is below the lower limit. Files that lie between the lower limit *LowPercentage* and the pre-migrate limit *PremigratePercentage* are copied and become dual-resident in both the internal GPFS storage pool and the designated external storage pool. This option allows the system to free up space quickly by deleting pre-migrated files if the pool becomes full. Specify a nonnegative integer in the range 0 to *LowPercentage*. The default is the same value as *LowPercentage*.

Notes:
1. *Percentage* values can be specified as numeric constants or floating-point values.
2. This option applies only when you migrate to the external storage pool.
3. This option does not apply when the current rule operates on one group pool.

THRESHOLD (*ResourceClass*)
Specifies the type of capacity-managed resources that are associated with *ListName*. The following values are valid:

FILESET_QUOTAS
Indicates that the **LIST** rule must use the occupancy percentage of the "hard limit" fileset quota per the **mmlsquota** and **mmedquota** commands.

FILESET_QUOTA_SOFT
Indicates that the **LIST** rule must use the occupancy percentage of the "soft limit" fileset quota per the **mmlsquota** and **mmedquota** commands.

GROUP_QUOTAS
  Indicates that the **LIST** rule must use the occupancy percentage of the "hard limit" group quota per the **mmlsquota** and **mmedquota** commands.

GROUP_QUOTA_SOFT
  Indicates that the **LIST** rule must use the occupancy percentage of the "soft limit" group quota per the **mmlsquota** and **mmedquota** commands.

POOL_CAPACITIES
  Indicates that the **LIST** rule uses the occupancy percentage of the pool when it applies the threshold rule. This value is the default value. This value is used if the threshold is not specified in the **EXTERNAL LIST** rule but appears in the **LIST** rule.

USER_QUOTAS
  Indicates that the **LIST** rule uses the occupancy percentage of the "hard limit" user quota per the **mmlsquota** and **mmedquota** commands.

USER_QUOTA_SOFT
  Indicates that the **LIST** rule uses the occupancy percentage of the "soft limit" user quota per the **mmlsquota** and **mmedquota** commands.

Note: This option does not apply when the current rule operates on one group pool.

For more detail on how **THRESHOLD** can be used to control file migration and deletion, see "Phase one: Selecting candidate files" on page 383 and "Pre-migrating files with external storage pools" on page 398.

TO POOL *ToPoolName*
Specifies the name of the storage pool to which all the files that match the rule criteria are migrated. This phrase is optional if the **COMPRESS** keyword is specified.

WEIGHT (*WeightExpression***)**
Establishes an order on the matching files. Specifies an SQL expression with a numeric value that can be converted to a double-precision floating point number. The expression can refer to any of the file attributes and can include any constants and any of the available SQL operators or built-in functions.

WHEN (*TimeBooleanExpression***)**
Specifies an SQL expression that evaluates to **TRUE** or **FALSE**, depending only on the SQL built-in variable **CURRENT_TIMESTAMP**. If the **WHEN** clause is present and *TimeBooleanExpression* evaluates to **FALSE**, the rule is skipped.

The **mmapplypolicy** command assigns the **CURRENT_TIMESTAMP** when it begins processing. It uses either the actual Coordinated Universal Time date and time or the date specified with the **-D** option.

WHERE *SqlExpression*
Specifies an SQL expression that can reference file attributes as SQL variables, functions, and operators. Some attributes are not available to all rules. Compares the file attributes specified in the rule with the attributes of the file that is created.

SqlExpression must be an expression that evaluates to **TRUE** or **FALSE**, but can be any combination of standard SQL syntax expressions, including built-in functions.

Omitting the **WHERE** clause entirely is equivalent to writing **WHERE TRUE**. The **WHERE** clause must be the last clause of the rule.

SQL expressions for policy rules
A number of the available clauses in the GPFS policy rules utilize SQL expressions.

You can reference different file attributes as SQL variables and combine them with SQL functions an operators. Depending on the clause, the SQL expression must evaluate to either **TRUE** or **FALSE**, a numeric value, or a character string. Not all file attributes are available to all rules.

File attributes in SQL expressions:

SQL expressions can include file attributes that specify certain clauses.

The following file attributes can be used in SQL expressions specified with the **WHERE**, **WEIGHT**, and **SHOW** clauses:

ACCESS_TIME
Specifies an SQL time stamp value for the date and time that the file was last accessed (POSIX atime). See **EXPIRATION_TIME**.

BLOCKSIZE
Specifies the size, in bytes, of each block of the file.

CHANGE_TIME
Specifies an SQL time stamp value for the date and time that the file metadata was last changed (POSIX ctime).

CLONE_DEPTH
Specifies the depth of the clone tree for the file.

CLONE_IS_PARENT
Specifies whether the file is a clone parent.

CLONE_PARENT_FILESETID
Specifies the fileset ID of the clone parent. The fileset ID is available only if **CLONE_PARENT_IS_SNAP** is a nonzero value.

CLONE_PARENT_INODE
Specifies the inode number of the clone parent, or **NULL** if it is not a file clone.

CLONE_PARENT_IS_SNAP
Specifies whether the clone parent is in a snapshot.

CLONE_PARENT_SNAP_ID
Specifies the snapshot ID of the clone parent. The snapshot ID is available only if **CLONE_PARENT_IS_SNAP** is a nonzero value.

CREATION_TIME
Specifies an SQL time stamp value that is assigned when a file is created.

DEVICE_ID
Specifies the ID of the device that contains the directory entry.

DIRECTORY_HASH
Can be used to group files within the same directory.

> **DIRECTORY_HASH** is a function that maps every **PATH_NAME** to a number. All files within the same directory are mapped to the same number and deeper paths are assigned to larger numbers.

> **DIRECTORY_HASH** uses the following functions:

CountSubstr(*BigString***,***LittleString***)**
Counts and returns the number of occurrences of *LittleString* in *BigString*.

HashToFloat(*StringValue***)**
Is a hash function that returns a quasi-random floating point number ≥ 0 and < 1, whose value depends on a string value. Although the result might appear random, **HashToFloat**(*StringValue*) always returns the same floating point value for a particular string value.

> The following rule lists the directory hash values for three directories:

```
RULE 'y' LIST 'xl' SHOW(DIRECTORY_HASH)
LIST 'xl' /abc/tdir/randy1 SHOW(+3.49449638091027E+000)
LIST 'xl' /abc/tdir/ax     SHOW(+3.49449638091027E+000)
```

```
LIST 'xl' /abc/tdir/mmPolicy.8368.765871DF/mm_tmp/PWL.12 SHOW(+5.21282524359412E+000)
LIST 'xl' /abc/tdir/mmPolicy.31559.1E018912/mm_tmp/PWL.3 SHOW(+5.10672733094543E+000)
LIST 'xl' /abc/tdir/mmPolicy.31559.1E018912/mm_tmp/PWL.2 SHOW(+5.10672733094543E+000)
```

The following rule causes files within the same directory to be grouped and processed together during deletion. Grouping the files can improve the performance of GPFS directory-locking and caching.

```
RULE 'purge' DELETE WEIGHT(DIRECTORY_HASH) WHERE (deletion-criteria)
```

EXPIRATION_TIME

Specifies the expiration time of the file, expressed as an SQL time-stamp value. If the expiration time of a file is not set, its expiration time is SQL NULL. You can detect such files by checking for "EXPIRATION_TIME IS NULL".

Remember the following points:

- EXPIRATION_TIME is tracked independently from ACCESS_TIME and both values are maintained for immutable files.
- Expiration time and indefinite retention are independent attributes. You can change the value of either one without affecting the value of the other.

FILE_HEAT

Specifies the heat of the file based on the file access time and access size. For more information, see /usr/lpp/mmfs/samples/ilm/README.

The calculation of **FILE_HEAT** depends partly on the value of the **atime** file attribute. The **-S** option of the **mmcrfs** and **mmchfs** commands controls whether and when **atime** is updated. You can override this setting temporarily with mount options that are specific to IBM Spectrum Scale. For more information, see the topics *mmchfs command* and *mmcrfs command* in the *IBM Spectrum Scale: Command and Programming Reference* and *atime values* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

FILE_SIZE

Specifies the current size or length of the file, in bytes.

FILESET_NAME

Specifies the fileset where the path name for the files is located, or is to be created.

Note: Using the **FOR FILESET** clause has the same effect and is more efficient to evaluate.

GENERATION

Specifies a number that is incremented whenever an INODE number is reused.

GROUP_ID

Specifies the numeric group ID of the file's group.

GROUP_NAME

Specifies the group name that is associated with **GROUP_ID**.

INODE

Specifies the file's inode number.

KB_ALLOCATED

Specifies the number of kilobytes of disk space allocated for the file data.

MODE

Displays the type and permission bits of a file as a 10-character string. The string has the same format as the first 10 characters of the output from the UNIX **ls -l** command. For example, **-rwxr-xr-x** is the **MODE** string of a file that can be read or executed by its owner, its group, or any user, but written only by its owner.

The first character of the **MODE** attributes displays the file type. The following values are supported:

d   Directory.

l   Link.

c   Character device.

b   Block device.

p   Pipe.

s   Socket.

?   Some other attribute, or unknown.

⌂

MISC_ATTRIBUTES
Specifies various miscellaneous file attributes. The value is a string of characters that are defined as follows:

+   File access is controlled by an Access Control List (ACL).

a   The file is appendOnly.

A   Archive.

c   The file is selected to be compressed.

D   Directory. To match all directories, you can use %D% as a wildcard character.

e   Encrypted. A Microsoft Windows file attribute. Does not refer to IBM Spectrum Scale file encryption.

E   The file has extended-attribute metadata.

f   Some data blocks of the file are ill-placed with respect to the File Placement Optimizer (FPO) attributes of the file.

F   Regular data file.

H   Hidden. A Microsoft Windows file attribute.

i   Not indexed by content. A Microsoft Windows file attribute.

I   Some data blocks might be ill-placed.

j   AFM append flag.

J   Some data blocks might be ill-replicated.

k   Remote attributes present. Internal to AFM.

K   Some data blocks might be ill-compressed.

L   Symbolic link.

m   Empty directory.

M   Co-managed.

2   Data blocks are replicated.

o   Offline.

O   Other (not F, D, nor L). For example, a device or named pipe.

p   Reparse point. A Microsoft Windows file attribute.

P   Active File Management (AFM) summary flag. Indicates that at least one specific AFM flag is set: **j**, **k**, **u**, **v**, **w**, **x**, **y**, or **z**.

r   Has streams. A Microsoft Windows file attribute.

R   Read-only.

s   Sparse. A Microsoft Windows file attribute.

S   System. A Microsoft Windows file attribute.

t   Temporary. A Microsoft Windows file attribute.

u   File is cached. Internal to AFM.

U   The file is **trunc**-managed.

v   AFM create flag.

V   Read-managed.

w   AFM dirty data flag.

W   Write-managed.

x   AFM hard-linked flag.

X   Immutability.

y   AFM attribute-changed flag.

Y   Indefinite retention.

z   AFM local flag.

Z   Secure deletion.

MODIFICATION_SNAPID
Specifies the integer ID of the snapshot after which the file was last changed. The value is normally derived with the **SNAP_ID()** built-in function that assigns integer values to GPFS snapshot names. This attribute allows policy rules to select files that are modified after a snapshot image is taken.

MODIFICATION_TIME
Specifies an SQL time stamp value for the date and time that the file data was last modified (POSIX mtime).

NAME
Specifies the name of a file.

NLINK
Specifies the number of hard links to the file.

PATH_NAME
Specifies a path for the file; the path includes the name of the file.

POOL_NAME
Specifies the storage pool where the file data is located.

> **Note:** Using the **FROM POOL** clause has the same effect and is often preferable.

SNAP_NAME
Specifies the snapshot name that the snapshot file is part of.

> **Note:** This attribute has an effect only when it is used in snapshot placement rules.

RDEVICE_ID
Specifies the device type for a device.

USER_ID
Specifies the numeric user ID of the owner of the file. To return the value of USER_ID when USER_NAME returns NULL, use COALESCE(USER_NAME, VARCHAR(USER_ID)) .

USER_NAME
Specifies the user name that is associated with **USER_ID**.

Notes:

1. When file attributes are referenced in initial placement rules, only the following attributes are valid: **CREATION_TIME,FILESET_NAME**, **GROUP_ID**, **MODE**, **NAME**, **SNAP_NAME**, and **USER_ID**. The placement rules, like all rules with a clause, might also reference the current date and current time and use them to control matching.

2. When file attributes are used for restoring files, the attributes correspond to the attributes at the time of the backup, not to the current restored file.

3. For SQL expressions, if you want to show any of these attribute fields as strings (for example, **FILE_HEAT**), use **SHOW('[FILE_HEAT]')** rather than **SHOW('FILE_HEAT')**, as the latter is expanded.

4. All date attributes are evaluated in Coordinated Universal Time (a time standard abbreviated as UTC).

Using built-in functions:

With GPFS, you can use built-in functions in comparison predicates, between predicates, in predicates, like predicates, mathematical value expressions, and boolean, string and numeric literals.

These functions are organized into the following categories:
- "Extended attribute functions"
- "String functions" on page 379
- "Numerical functions" on page 380
- "Date and time functions" on page 381

Extended attribute functions:

You can use these functions to support access to the extended attributes of a file, and to support conversion of values to the supported SQL data types.

The following attribute functions can be used:

GetXattrs(*pattern***,***prototype***)**

Returns extended attribute **key=value** pairs of a file for all extended attributes whose keys that match *pattern*. The **key=value** pairs are returned in the format specified by *prototype*.

If the value specified for *pattern* is '*' or empty then all keys are matched.

The *prototype* is a character string representing the format of a typical **key=value** pair. The *prototype* allows the user to specify which characters will be used to quote values, escape special code points, separate the key and value, and separate each **key=value** pair.

Some examples of the *prototype* argument include:

```
key~n=value^n,      # specify the escape characters
hexkey=hexvalue,    # specify either or both as hexadecimal values
"key\n"="value\n",  # specify quotes on either or both
key:"value^n";      # specify alternatives to = and ,
k:"v^n";            # allow key and value to be abbreviated
key,                # specify keys only
"value~n";          # specify values only
key='value~n'&      # alternative quoting character
key=value           # do not use a ',' separator; use space instead
```

You may omit the last or both arguments. The defaults are effectively **GetXattrs('*','key^n=hexvalue,')**.

The **GetXattrs** function returns an empty string for files that have no extended attributes with keys that match *pattern*.

The **GetXattrs** function is supported by the **mmapplypolicy** command, but it might return **NULL** in other contexts.

SetBGF(*BlockGroupFactor***)**

Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if **--allow-write-affinity** is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

SetWAD(*WriteAffinityDepth***)**

Specifies the allocation policy to be used. This option only works if **--allow-write-affinity** is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

SetWADFG("*WadfgValueString***")**

Indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in the file are to be written. You use this parameter to determine the layout of a file in the cluster so as to optimize the typical access patterns of your applications. This applies only to a new data block layout; it does not migrate previously existing data blocks.

"*WadfgValueString*" is a semicolon-separated string identifying one or more failure groups in the following format:

```
FailureGroup1[;FailureGroup2[;FailureGroup3]]
```

where each *FailureGroupx* is a comma-separated string identifying the rack (or range of racks), location (or range of locations), and node (or range of nodes) of the failure group in the following format:

```
Rack1{:Rack2{:...{:Rackx}}},Location1{:Location2{:...{:Locationx}}},ExtLg1{:ExtLg2{:...{:ExtLgx}}}
```

For example, the following value

```
1,1,1:2;2,1,1:2;2,0,3:4
```

means that the first failure group is on rack 1, location 1, extLg 1 or 2; the second failure group is on rack 2, location 1, extLg 1 or 2; and the third failure group is on rack 2, location 0, extLg 3 or 4.

If the end part of a failure group string is missing, it is interpreted as 0. For example, the following are interpreted the same way:

```
2
2,0
2,0,0
```

Notes:

1. Only the end part of a failure group string can be left off. The missing end part may be the third field only, or it may be both the second and third fields; however, if the third field is provided, the second field must also be provided. The first field must *always* be provided. In other words, every comma must both follow and precede a number; therefore, *none* of the following are valid:

```
2,0,
2,
,0,0
0,,0
,,0
```

2. Wildcard characters (*) are supported in these fields.

Here is an example of using setBGF, setWAD, and setWADFG:

```
RULE 'bgf' SET POOL 'pool1' WHERE NAME LIKE '%' AND setBGF(128) AND setWAD(1) AND setWADFG(1,0,1;2,0,1;3,0,1)
```

This rule has the same effect as the following command:

```
mmchattr --block-group-factor 128 --write-affinity-depth 1 --write-affinity-failuregroup "1,0,1;2,0,1;3,0,1"  test
```

After installing this policy, a newly created file will have the same values for these three extended attributes as it would if **mmchattr** were used to set them:

```
(06:29:11) hs22n42:/sncfs # mmlsattr -L test
file name:            test
metadata replication: 3 max 3
data replication:    3 max 3
immutable:           no
appendOnly:          no
flags:
storage pool name:    system
fileset name:        root
snapshot name:
Block group factor:  128                                       ----------------gpfs.BGF
Write affinity depth: 1                                        ----------------gpfs.WAD
Write Affinity Depth Failure Group(FG) Map for copy:1 1,0,1    ----------------gpfs.WADFG
Write Affinity Depth Failure Group(FG) Map for copy:2 2,0,1
Write Affinity Depth Failure Group(FG) Map for copy:3 3,0,1
creation time:       Sat Jun  8 06:28:50 2013
Misc attributes:     ARCHIVE
```

SetXattr('*ExtendedAttributeName*', '*ExtendedAttributeValue*')
> This function sets the value of the specified extended attribute of a file.

> Successful evaluation of **SetXattr** in a policy rule returns the value **TRUE** and sets the named extended attribute to the specified value for the file that is the subject or object of the rule. This function is effective for policy rules (like **MIGRATE** and **LIST**) that are evaluated by **mmapplypolicy** and for the policy placement rule, **SET POOL**, when a data file is about to be created.

XATTR(*extended-attribute-name* [, *start* [, *length*]])
> Returns the value of a substring of the extended attribute that is named by its argument as an SQL VARCHAR value, where:

> *extended-attribute-name*
>> Specifies any SQL expression that evaluates to a character string value. If the named extended attribute does not exist, **XATTR** returns the special SQL value **NULL**.

>> **Note:** In SQL, the expression **NULL** || *AnyValue* yields **NULL**. In fact, with a few exceptions, the special SQL value of **NULL** "propagates" throughout an SQL expression, to yield **NULL**. A notable exception is that *(expression)* **IS NULL** always yields either **TRUE** or **FALSE**, never **NULL**.

>> For example, if you wish to display a string like _NULL_ when the value of the extended attribute of a file is **NULL** you will need to code your policy rules file like this:
>> ```
>> define(DISPLAY_NULL,[COALESCE($1,'_NULL_')])
>> rule external list 'a' exec ''
>> rule list 'a' SHOW(DISPLAY_NULL(xattr('user.marc')) || ' and ' || DISPLAY_NULL(xattr('user.eric')))
>> ```

>> Here is an example execution, where either or both of the values of the two named extended attributes may be **NULL**:
>> ```
>> mmapplypolicy /gig/sill -P /ghome/makaplan/policies/display-null.policy -I test -L 2
>>    ...
>> WEIGHT(inf) LIST 'a' /gg/sll/cc SHOW(_NULL_ and _NULL_)
>> WEIGHT(inf) LIST 'a' /gg/sll/mm SHOW(yes-marc and _NULL_)
>> WEIGHT(inf) LIST 'a' /gg/sll/bb SHOW(_NULL_ and yes-eric)
>> WEIGHT(inf) LIST 'a' /gg/sll/tt SHOW(yes-marc and yes-eric)
>> ```

>> GPFS/Policy/SQL is a subset of standard ISO/ANSI SQL, with additional extensions and modifications to facilitate GPFS/ILM. Regarding the **NULL** value, GPFS/Policy/SQL supports the "unknown value" meaning of **NULL**.

> *start*
>> Is the optional starting position within the extended attribute value. The default is 1.

> *length*
>> Is the optional length, in bytes, of the extended attribute value to return. The default is the number of bytes from the start to the end of the extended attribute string.

Note: **XATTR** (*name,i,k*) == **SUBSTR**(**XATTR**(*name*),*i,k*).

Some extended attribute values represent numbers or timestamps as decimal or binary strings. Use the **TIMESTAMP**, **XATTR_FLOAT**, or **XATTR_INTEGER** function to convert extended attributes to SQL numeric or timestamp values:

XATTR_FLOAT(*extended-attribute-name* **[,** *start* **[,** *length***, [,** *conversion_option***]]])**
> Returns the value of a substring of the extended attribute that is named by its argument, converted to an SQL double floating-point value, where:

> *extended-attribute-name*
>> Specifies any SQL expression that evaluates to a character string value. If the named extended attribute does not exist, **XATTR** returns the special SQL value **NULL**.

> *start*
>> Is the optional starting position within the extended attribute value. The default is 1.

> *length*
>> Is the optional length, in bytes, of the extended attribute value to return. The default is the number of bytes from the start to the end of the extended attribute string. You can specify length as -1 to reach from the start to the end of the extended attribute string.

> *conversion_option*
>> Specifies how the bytes are to be converted to a floating-point value. Supported options include:
>> - BIG_ENDIAN_DOUBLE or BD - a signed binary representation, IEEE floating, sign + 11 bit exponent + fraction. This is the default when executing on a "big endian" host OS, such as AIX on PowerPC®.
>> - BIG_ENDIAN_SINGLE or BS - IEEE floating, sign + 8-bit exponent + fraction.
>> - LITTLE_ENDIAN_DOUBLE or LD - bytewise reversed binary representation. This is the default when executing on a "little endian" host OS, such as Linux on Intel x86.
>> - LITTLE_ENDIAN_SINGLE or LS - bytewise-reversed binary representation.
>> - DECIMAL - the conventional SQL character string representation of a floating-point value.

>> **Notes:**
>> 1. Any prefix of a conversion name can be specified instead of spelling out the whole name. The first match against the list of supported options is used; for example, L matches LITTLE_ENDIAN_DOUBLE.
>> 2. If the extended attribute does not exist, the selected substring has a length of 0, or the selected bytes cannot be converted to a floating-point value, the function returns the special SQL value **NULL**.

XATTR_INTEGER(*extended-attribute-name* **[,** *start* **[,** *length***, [,** *conversion_option***]]])**
> Returns the value of (a substring of) the extended attribute named by its argument, converted to a SQL LARGEINT value, where.

> *extended-attribute-name*
>> Specifies any SQL expression that evaluates to a character string value. If the named extended attribute does not exist, **XATTR** returns the special SQL value **NULL**.

> *start*
>> Is the optional starting position within the extended attribute value. The default is 1.

> *length*
>> Is the optional length, in bytes, of the extended attribute value to return. The default is the number of bytes from the start to the end of the extended attribute string. You can specify length as -1 to reach from the start to the end of the extended attribute string.

> *conversion_option*
>> Specifies how the bytes are to be converted to a LARGEINT value. Supported options include:

- BIG_ENDIAN - a signed binary representation, most significant byte first. This is the default when executing on a "big endian" host OS, such as AIX on PowerPC.
- LITTLE_ENDIAN - bytewise reversed binary representation. This is the default when executing on a "little endian" host OS, such as Linux on Intel x86.
- DECIMAL - the conventional SQL character string representation of an integer value.

Notes:
1. Any prefix of a conversion name can be specified instead of spelling out the whole name (B, L, or D, for example).
2. If the extended attribute does not exist, the selected substring has a length of 0, or the selected bytes cannot be converted to a LARGEINT value, the function returns the special SQL value **NULL**. For example:
   ```
   XATTR_INTEGER('xyz.jim',5,-1,'DECIMAL')
   ```

String functions:

You can use these string-manipulation functions on file names and literal values.

Important tips:
1. You must enclose strings in single-quotation marks.
2. You can include a single-quotation mark in a string by using two single-quotation marks. For example, **'a''b'** represents the string **a'b**.

CHAR(*expr*[, *length*])
Returns a fixed-length character string representation of its *expr* argument, where:

expr
   Can be any data type.

length
   If present, must be a literal, integer value.

The resulting type is **CHAR** or **VARCHAR**, depending upon the particular function called.

The string that **CHAR** returns is padded with blanks to fill the *length* of the string. If *length* is not specified, it defaults to a value that depends on the type of the argument (*expr*).

Note: The maximum length of a **CHAR** (fixed length string) value is 255 bytes. The result of evaluating an SQL expression whose result is type **CHAR** may be truncated to this maximum length.

CONCAT(*x*,*y*)
Concatenates strings *x* and *y*.

HEX(*x*)
Converts an integer *x* into hexadecimal format.

LENGTH(*x*)
Determines the length of the data type of string *x*.

LOWER(*x*)
Converts string *x* into lowercase.

REGEX(*String*,'*Pattern*')
Returns **TRUE** if the pattern matches, **FALSE** if it does not. *Pattern* is a Posix extended regular expression.

Note: The policy SQL parser normally performs M4 macro preprocessing with square brackets set as the quote characters. Therefore, it is recommended that you add an extra set of square brackets around your **REGEX** pattern string; for example:

```
...WHERE REGEX(name,['^[a-z]*$'])  /* only accept lowercase alphabetic file names */
```

The following SQL expression:

```
NOT REGEX(STRING_VALUE,['^[^z]*$|^[^y]*$|^[^x]*$|[abc]'])
```

can be used to test if STRING_VALUE contains *all* of the characters x, y, and z, in any order, and *none* of the characters a, b, or c.

REGEXREPLACE(*string***,***pattern***,***result-prototype-string***)**
Returns a character string as *result-prototype-string* with occurrences of \\i (where i is 0 through 9) replaced by the substrings of the original string that match the ith parenthesis delimited parts of the pattern string. For example:

```
REGEXREPLACE('speechless',['([^aeiou]*)([aeiou]*)(.*)'],['last=\3. middle=\2. first=\1.'])
```

returns the following:

```
'last=chless. middle=ee. first=sp.'
```

When *pattern* does not match *string*, **REGEXREPLACE** returns the value **NULL**.

When a **\\0** is specified in the *result-prototype-string*, it is replaced by the substring of *string* that matches the entire *pattern*.

SUBSTR(*x***,***y***,***z***)**
Extracts a portion of string x, starting at position y, optionally for z characters (otherwise to the end of the string). This is the short form of **SUBSTRING**. If y is a negative number, the starting position is counted from the end of the string; for example, **SUBSTR('ABCDEFGH',-3,2)** == 'FG'.

Note: Do not confuse **SUBSTR** with **substr**. **substr** is an m4 built-in macro function.

SUBSTRING(*x* **FROM** *y* **FOR** *z***)**
Extracts a portion of string x, starting at position y, optionally for z characters (otherwise to the end of the string).

UPPER(*x***)**
Converts the string x into uppercase.

VARCHAR(*expr* **[,** *length* **])**
Returns a varying-length character string representation of a character string, date/time value, or numeric value, where:

expr
    Can be any data type.

length
    If present, must be a literal, integer value.

The resulting type is **CHAR** or **VARCHAR**, depending upon the particular function called. Unlike **CHAR**, the string that the **VARCHAR** function returns is not padded with blanks.

Note: The maximum length of a **VARCHAR**(variable length string) value is 8192 bytes. The result of evaluating an SQL expression whose result is type **VARCHAR** may be truncated to this maximum length.

Numerical functions:

You can use numeric-calculation functions to place files based on either numeric parts of the file name, numeric parts of the current date, or UNIX-client user IDs or group IDs.

These functions can be used in combination with comparison predicates and mathematical infix operators (such as addition, subtraction, multiplication, division, modulo division, and exponentiation).

INT(*x*)
Converts number *x* to a whole number, rounding up fractions of .5 or greater.

INTEGER(*x*)
Converts number *x* to a whole number, rounding up fractions of .5 or greater.

MOD(*x*,*y*)
Determines the value of *x* taken modulo *y* (*x* % *y*).

Date and time functions:

You can use these date-manipulation and time-manipulation functions to place files based on when the files are created and the local time of the GPFS node serving the directory where the file is being created.

CURRENT_DATE
Determines the current date on the GPFS server.

CURRENT_TIMESTAMP
Determines the current date and time on the GPFS server.

DAYOFWEEK(*x*)
Determines the day of the week from date or timestamp *x*. The day of a week is from 1 to 7 (Sunday is 1).

DAYOFYEAR(*x*)
Determines the day of the year from date *x*. The day of a year is a number from 1 to 366.

DAY(*x*)
Determines the day of the month from date or timestamp *x*.

DAYS(*x*)
Determines the number of days between date or timestamp *x* and 0001-01-01.

DAYSINMONTH(*x*)
Determines the number of days in the month of date *x*.

DAYSINYEAR(*x*)
Determines the day of the year of date *x*.

HOUR(*x*)
Determines the hour of the day (a value from 0 to 23) of timestamp *x*.

MINUTE(*x*)
Determines the minute from timestamp *x*.

MONTH(*x*)
Determines the month of the year from date or timestamp *x*.

QUARTER(*x*)
Determines the quarter of year from date *x*. Quarter values are the numbers 1 through 4. For example, January, February, and March are in quarter 1.

SECOND(*x*)
Returns the seconds portion of timestamp *x*.

TIMESTAMP(*sql-numeric-value*) or TIMESTAMP(*sql-character-string-value*)
Accepts any numeric value. The numeric value is interpreted as the number of seconds since January 1, 1970 (the standard UNIX epoch) and is converted to an SQL TIMESTAMP value.

Signed 64-bit LARGEINT argument values are supported. Negative argument values cause **TIMESTAMP** to convert these values to timestamps that represent years before the UNIX epoch.

This function also accepts character strings of the form *YYYY-MM-DD HH:MM:SS*. A hyphen (-) or an at sign (@) might appear instead of the blank between the date and the time. The time can be omitted. An omitted time defaults to 00:00:00. The *:SS* field can be omitted, which defaults to 00.

WEEK(*x***)**
    Determines the week of the year from date *x*.

YEAR(*x***)**
    Determines the year from date or timestamp *x*.

All date and time functions use Universal Time (UT).

Example of a policy rules file
```
/*
Sample GPFS policy rules file
*/

rule 'vip' set pool 'pool0' where USER_ID <= 100
RULE 'm1' SET POOL 'pool1' WHERE LOWER(NAME) LIKE '%marc%'
RULE SET POOL 'pool1' REPLICATE (2) WHERE UPPER(NAME) = '%IBM%'
RULE 'r2' SET POOL 'pool2' WHERE UPPER(SUBSTRING(NAME FROM 1 FOR 4)) = 'GPFS'
RULE 'r3' SET POOL 'pool3' WHERE LOWER(SUBSTR(NAME,1,5)) = 'roger'
RULE SET POOL 'pool4' WHERE LENGTH(NAME) = 7
RULE SET POOL 'pool5' WHERE name like 'xyz%' AND name like '%qed' OR name like '%.tmp%'
RULE SET POOL 'pool6' WHERE name like 'abc%' OR name like '%xyz' AND name like 'x%'

RULE 'restore' RESTORE TO POOL 'pool0' where USER_ID <= 100


/* If none of the rules matches put those files in system pool */
rule 'default' SET POOL 'system'
```

Miscellaneous SQL functions:

The following miscellaneous SQL functions are available.

SNAP_ID([*'FilesetName'***,]** ***'SnapshotName'***)**
    Given an (optional) fileset/inode-space name and a snapshot name, this function returns the numeric snapshot ID of the given snapshot within the given inode-space.

GetEnv(*'EnvironmentVariableName'***)**
    This function gets the value of the specified environment variable.

GetMMconfig(*'GPFSConfigurationParameter'***)**
    This function gets the value of the specified GPFS configuration parameter.

The mmapplypolicy command and policy rules

The **mmapplypolicy** command has policy rules that are based on the characteristics of different phases.

Any given file is a potential candidate for at most one **MIGRATE** or **DELETE** operation during each invocation of the **mmapplypolicy** command. A single invocation of the **mmapplypolicy** command is called the *job*.

The **mmapplypolicy** command sets the SQL built-in variable **CURRENT_TIMESTAMP**, and collects pool occupancy statistics at the beginning of the job.

The **mmapplypolicy** job consists of three major phases:
1. "Phase one: Selecting candidate files" on page 383
2. "Phase two: Choosing and scheduling files" on page 384
3. "Phase three: Migrating and deleting files" on page 386

Phase one: Selecting candidate files

In the first phase of the **mmapplypolicy** job, all the files within the specified GPFS file system device, or below the input path name, are scanned. The attributes of each file are read from the file's GPFS inode structure.

Note: mmapplypolicy reads directly from the metadata disk blocks and can therefore lag behind the posix state of the file system. To be sure that `MODIFICATION_TIME` and the other timestamps are completely up to date, you can use the following suspend-and-resume sequence to force recent changes to disk:

```
mmfsctl fs-name suspend; mmfsctl fs-name resume;
```

For each file, the policy rules are considered, in order, from first rule to last:

*   If the rule has a **WHEN** clause that evaluates to **FALSE**, the rule is skipped.
*   If the rule has a **FROM POOL** clause, and the named pool does not match the **POOL_NAME** attribute of the file, the rule is skipped. A **FROM POOL** clause that specifies a group pool name matches a file if any pool name within the group pool matches the **POOL_NAME** attribute of the file.
*   If there is a **THRESHOLD** clause and the current pool of the file has an occupancy percentage that is less than the *HighPercentage* parameter of the **THRESHOLD** clause, the rule is skipped.
*   If the rule has a **FOR FILESET** clause, but none of the named filesets match the **FILESET_NAME** attribute of the file, the rule is skipped.
*   If the rule has a **WHERE** clause that evaluates to **FALSE**, the rule is skipped. Otherwise, the rule applies.
*   If the applicable rule is a **LIST** '*listname-y*' rule, the file becomes a candidate for inclusion in the named list unless the **EXCLUDE** keyword is present, in which case the file will not be a candidate; nor will any following **LIST** '*listname-y*' rules be considered for the subject file. However, the file is subject to **LIST** rules naming other list names.
*   If the applicable rule is an **EXCLUDE** rule, the file will be neither migrated nor deleted. Files matching the **EXCLUDE** rule are not candidates for any **MIGRATE** or **DELETE** rule.

    Note: Specify the **EXCLUDE** rule before any other rules that might match the files that are being excluded. For example:

    ```
    RULE 'Exclude root's file' EXCLUDE where USER_ID = 0
    RULE 'Migrate all but root's files' MIGRATE TO POOL 'pool1'
    ```

    will migrate all the files that are not owned by **root**. If the **MIGRATE** rule was placed in the policy file before the **EXCLUDE** rule, all files would be migrated because the policy engine would evaluate the rules from first to last, and **root**'s files would have to match the **MIGRATE** rule.

    To exclude files from matching a **LIST** rule, you must create a separate **LIST** rule with the **EXCLUDE** clause and place it before the **LIST** rule.
*   If the applicable rule is a **MIGRATE** rule, the file becomes a *candidate* for migration to the pool specified by the **TO POOL** clause.

    When a group pool is the **TO POOL** target of a **MIGRATE** rule, the selected files are distributed among the disk pools comprising the group pool, with files of highest weight going to the most preferred disk pool up to the occupancy limit for that pool. If there are still more files to be migrated, those go to the second most-preferred pool up to the occupancy limit for that pool (again choosing the highest-weight files from among the remaining selected files); and so on for the subsequent most-preferred pools, until either all selected files have been migrated or until all the disk pools of the group pool have been filled to their respective limits.
*   If the applicable rule is a **DELETE** rule, the file becomes a *candidate* for deletion.
*   If there is no applicable rule, the file is not a candidate for migration or deletion.
*   Each candidate file (for migration or deletion) is also associated with a *LowPercentage* occupancy percentage value, which is taken from the **THRESHOLD** clause of the applicable rule. If not specified, the *LowPercentage* value defaults to 0%.

- Each candidate file is also associated with a numeric *weight*, either computed from the *WeightExpression* of the applicable rule, or assigned a default using these rules:
  - If a *LowPercentage* is specified within a **THRESHOLD** clause of the applicable rule, the weight of the candidate is taken as the **KB_ALLOCATED** attribute of the candidate file.
  - If a *LowPercentage* is not specified within a **THRESHOLD** clause of the applicable rule, the weight of the candidate is taken as **+infinity**.

Phase two: Choosing and scheduling files

In the second phase of the **mmapplypolicy** job, some or all of the candidate files are chosen.

Chosen files are scheduled for migration or deletion, taking into account the weights and thresholds determined in "Phase one: Selecting candidate files" on page 383, as well as the actual pool occupancy percentages. Generally, candidates with higher weights are chosen ahead of those with lower weights.

File migrations to and from external pools are done before migrations and deletions that involve only GPFS disk pools.

File migrations that do not target group pools are done before file migrations to group pools.

File migrations that target a group pool are done so that candidate files with higher weights are migrated to the more preferred GPFS disk pools within the group pool, but respecting the **LIMIT**s specified in the group pool definition.

The following two options can be used to adjust the method by which candidates are chosen:

--choice-algorithm {best | exact | fast}
   Specifies one of the following types of algorithms that the policy engine is to use when selecting candidate files:

   best
      Chooses the optimal method based on the rest of the input parameters.

   exact
      Sorts all of the candidate files completely by weight, then serially considers each file from highest weight to lowest weight, choosing feasible candidates for migration, deletion, or listing according to any applicable rule **LIMIT**s and current storage-pool occupancy. This is the default.

   fast
      Works together with the parallelized **-g /shared-tmp -N** node-list selection method. The **fast** choice method does not completely sort the candidates by weight. It uses a combination of statistical, heuristic, and parallel computing methods to favor higher weight candidate files over those of lower weight, but the set of chosen candidates may be somewhat different than those of the **exact** method, and the order in which the candidates are migrated, deleted, or listed is somewhat more random. The **fast** method uses statistics gathered during the policy evaluation phase. The **fast** choice method is especially fast when the collected statistics indicate that either all or none of the candidates are feasible.

--split-margin *n.n*
   A floating-point number that specifies the percentage within which the **fast**-choice algorithm is allowed to deviate from the **LIMIT** and **THRESHOLD** targets specified by the policy rules. For example if you specified a **THRESHOLD** number of 80% and a split-margin value of 0.2, the **fast**-choice algorithm could finish choosing files when it reached 80.2%, or it might choose files that bring the occupancy down to 79.8%. A nonzero value for split-margin can greatly accelerate the execution of the **fast**-choice algorithm when there are many small files. The default is 0.2.

File grouping and the SIZE clause

When scheduling files, **mmapplypolicy** simply groups together either the next 100 files by default, or the number of files explicitly set using the **-B** option.

However, you can set up **mmapplypolicy** to schedule files so that each invocation of the InterfaceScript gets approximately the same amount of file data to process. To do so, use the **SIZE** clause of certain policy rules to specify that scheduling be based on the sum of the sizes of the files. The **SIZE** clause can be applied to the following rules (for details, see "Policy rules" on page 363):
- **DELETE**
- **EXTERNAL LIST**
- **EXTERNAL POOL**
- **LIST**
- **MIGRATE**

Administrator-specified customized file grouping or aggregation

In addition to using the **SIZE** clause to control the *amount* of work passed to each invocation of a InterfaceScript, you can also specify that files with *similar attributes* be grouped or aggregated together during the scheduling phase. To do so, use an aggregator program to take a list of chosen candidate files, sort them according to certain attributes, and produce a reordered file list that can be passed as input to the user script.

You can accomplish this by following these steps:
1. Run **mmapplypolicy** with the **-I prepare** option to produce a list of chosen candidate files, but not pass the list to a InterfaceScript.
2. Use your aggregator program to sort the list of chosen candidate files into groups with similar attributes and write each group to a new, separate file list.
3. Run **mmapplypolicy** with the **-r** option, specifying a set of file list files to be read. When invoked with the **-r** option, **mmapplypolicy** does not choose candidate files; rather, it passes the specified file lists as input to the InterfaceScript.

   Note: You can also use the **-q** option to specify that small groups of files are to be taken in round-robin fashion from the input file lists (for example, take a small group of files from x.list.A, then from x.list.B, then from x.list.C, then back to x.list.A, and so on, until all of the files have been processed).

   To prevent **mmapplypolicy** from redistributing the grouped files according to size, omit the **SIZE** clause from the appropriate policy rules and set the bunching parameter of the **-B** option to a very large value.

Reasons for candidates not to be chosen for deletion or migration

Generally, a candidate is not chosen for deletion from a pool, nor migration out of a pool, when the pool occupancy percentage falls below the *LowPercentage* value. Also, candidate files will not be chosen for migration into a target **TO POOL** when the target pool reaches the occupancy percentage specified by the **LIMIT** clause (or 99% if no **LIMIT** was explicitly specified by the applicable rule).

The limit clause does not apply when the target **TO POOL** is a group pool; the limits specified in the rule defining the target group pool govern the action of the **MIGRATE** rule. The policy-interpreting program (for example, **mmapplypolicy**) may issue a warning if a **LIMIT** clause appears in a rule whose target pool is a group pool.

Phase three: Migrating and deleting files

In the third phase of the **mmapplypolicy** job, the candidate files that were chosen and scheduled by the second phase are migrated or deleted, each according to its applicable rule.

For migrations, if the applicable rule had a **REPLICATE** clause, the replication factors are also adjusted accordingly. It is acceptable for the effective **FROM POOL** and **TO POOL** to be the same because the **mmapplypolicy** command can be used to adjust the replication factors of files without necessarily moving them from one pool to another.

The migration performed in the third phase can involve large amounts of data movement. Therefore, you may want to consider using the **–I defer** option of the **mmapplypolicy** command, and then perform the data movements with the **mmrestripefs -p** command.

Policy rules: Examples and tips

Before you write and apply policies, consider the following advice.

It is a good idea to test your policy rules by running the **mmapplypolicy** command with the **-I test** option and the **-L 3** or higher option. Testing helps you understand which files are selected as candidates and which candidates are chosen to be processed.

Do not apply a policy to an entire file system of important files unless you are confident that the rules correctly express your intentions. To test your rules, choose a directory that contains a small number of files, some of which you expect to be selected by your SQL policy rules and some of which you expect to be skipped.

Then enter a command like the following one:

```
mmapplypolicy /TestSubdirectory  -L 6  -I test
```

The output shows which files are scanned and which match rules or no rules. If a problem is not apparent, you can add a SHOW() clause to your rule to see the values of file attributes or SQL expressions. To see multiple values, enter a command like the following one:

```
SHOW('x1=' || varchar(Expression1) || ' x2='  || varchar(Expression2) || ... )
```

where *ExpressionX* is the SQL variable or expression of function that you suspect or do not understand. Beware that if any expression evaluates to SQL NULL, then the entire show clause is NULL, by the rules of SQL. One way to show null vs. non-null values is to define a macro and call it as in the following example:

```
define(DISPLAY_NULL,[CASE WHEN ($1) IS NULL THEN '_NULL_' ELSE varchar($1) END])
```

```
rule list a SHOW( 'x1=' || DISPLAY_NULL(xattr('user.marc')) || ' and  x2=' || DISPLAY_NULL(xattr('user.eric')))
```

Note: For examples and more information on the **-L** flag, see the topic *The mmapplypolicy -L command* in the *IBM Spectrum Scale: Problem Determination Guide*.

The following examples illustrate some useful policy rule techniques:

1. Delete files from the storage pool that is named **pool_1** that were not accessed in the last 30 days and that are named like temporary files or appear in any directory that is named **tmp**:

   ```
   RULE 'del1' DELETE FROM POOL 'pool_1'
      WHERE (DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME) > 30)
           AND (lower(NAME) LIKE '%.tmp' OR PATH_NAME LIKE '%/tmp/%')
   ```

2. Use the SQL **LIKE** predicate to test file names and path names:

   ```
   RULE '*/_*'  DELETE WHERE PATH_NAME LIKE '%/x_%' ESCAPE 'x'
   RULE '*XYZ*' DELETE WHERE NAME LIKE '%XYZ%'
   RULE '12_45' DELETE WHERE NAME LIKE '12x_45' ESCAPE 'x'
   RULE '12%45' DELETE WHERE NAME LIKE '12x%45' ESCAPE 'x'
   ```

```
RULE '12?45' DELETE WHERE NAME LIKE '12_45'
RULE '12*45' DELETE WHERE NAME LIKE '12%45'
RULE '*_*'   DELETE WHERE NAME LIKE '%x_%' ESCAPE 'x'
```

Where:

- A percent **%** wildcard in the name represents zero or more characters.
- An underscore (_) wildcard in the name represents 1 byte.

Use the optional **ESCAPE** clause to establish an escape character, when you need to match '_' or '%' exactly.

3. Use the SQL **UPPER** and **LOWER** functions to ignore case when testing names:

```
RULE 'UPPER'  DELETE WHERE upper(PATH_NAME) LIKE '%/TMP/OLD/%'
RULE 'lower'  DELETE WHERE lower(PATH_NAME) LIKE '%/tmp/old/%'
```

4. Use the SQL **SUBSTR** or **SUBSTRING** functions to test a substring of a name:

```
RULE 's1' DELETE WHERE SUBSTRING(NAME FROM 1 FOR 5)='XXXX-'
RULE 's2' DELETE WHERE SUBSTR(NAME,1,5)='YYYY-'
```

5. Use the SQL **SUBSTR** and **LENGTH** functions to test the suffix of a name:

```
RULE 'sfx' DELETE WHERE SUBSTR(NAME,LENGTH(NAME)-3)='.tmp'
```

6. Use a **WHEN** clause to restrict rule applicability to a particular day of the week:

```
RULE 'D_SUN' WHEN (DayOfWeek(CURRENT_DATE)=1) /* Sunday */
  DELETE WHERE PATH_NAME LIKE '%/tmp/%'
```

CURRENT_DATE is an SQL built in operand that returns the date portion of the **CURRENT_TIMESTAMP** value.

7. Use the SQL **IN** operator to test several possibilities:

```
RULE 'D_WEEKEND' WHEN (DayOfWeek(CURRENT_DATE) IN (7,1)) /*  Saturday or Sunday */
  DELETE WHERE PATH_NAME LIKE '%/tmp/%'
```

For information on how to use a macro processor such as **m4** to make reading and writing policy rules easier, see "Using macro processing utilities with policy rules" on page 390.

8. Use a **FILESET** clause to restrict the rule to files within particular filesets:

```
RULE 'fsrule1' MIGRATE TO POOL 'pool_2'
        FOR FILESET('root','fset1')
```

In this example there is no **FROM POOL** clause, so regardless of their current storage pool placement, all files from the named filesets are subject to migration to storage pool **pool_2**.

Note: To have the migrate rule applied to snapshot files, you must specify the `mmapplypolicy fs -S snap1` option, where `snap1` is the name of the snapshot where the files reside.

9. Use an **EXCLUDE** rule to exclude a set of files from all subsequent rules:

```
RULE 'Xsuper' EXCLUDE WHERE USER_ID=0
RULE 'mpg'    DELETE  WHERE lower(NAME) LIKE '%.mpg' AND FILE_SIZE>20123456
```

Notes:

a. Specify the **EXCLUDE** rule before rules that might match the files that are being excluded.

b. You cannot define a list and what to exclude from the list in a single rule. You must define two LIST statements, one specifying which files are in the list and one specifying what to exclude from the list. For example, to exclude files that contain the word **test** from the LIST rule **allfiles**, define the following rules:

```
RULE EXTERNAL LIST 'allfiles' EXEC '/u/brownap/policy/CHE/exec.list'

RULE 'exclude_allfiles' LIST 'allfiles' EXCLUDE where name like '%test%'

RULE 'all' LIST 'allfiles' SHOW('misc_attr ='|| MISC_ATTRIBUTES || HEX(MISC_ATTRIBUTES)) \
   where name like '%'
```

10. Use the SQL **NOT** operator with keywords, along with **AND** and **OR**:

```
RULE 'D_WEEKDAY' WHEN (DayOfWeek(CURRENT_DATE) NOT IN (7,1)) /*  a weekday */
  DELETE WHERE (PATH_NAME LIKE '%/tmp/%' OR NAME LIKE '%.tmp')
          AND (KB_ALLOCATED > 9999 AND NOT USER_ID=0)
```

11. To migrate only snapshot files that for which data blocks are allocated, use the following rule:

```
RULE "migrate snap data" MIGRATE FROM POOL X TO POOL Y WHERE KB_ALLOCATED > 0
```

12. Use a **REPLICATE** clause to increase the availability of selected files:

```
RULE 'R2' MIGRATE FROM POOL 'ypooly' TO POOL 'ypooly'
  REPLICATE(2) WHERE USER_ID=0
```

Before you increase the data replication factor for any file, the file system must be configured to support data replication.

13. The difference of two SQL Timestamp values can be compared to an SQL Interval value:

```
rule 'a' migrate to pool 'A' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' DAYS
rule 'b' migrate to pool 'B' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' HOURS
rule 'c' migrate to pool 'C' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' MINUTES
rule 'd' migrate to pool 'D' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' SECONDS
```

For the best precision, use the INTERVAL...SECONDS construct.

14. By carefully assigning both weights and thresholds, the administrator can formally express rules like this:

If the storage pool named **pool_X** has an occupancy percentage greater than 90% now, bring the occupancy percentage of storage pool that is named **pool_X** down to 80% by migrating files that are three months or older to the storage pool that is named **pool_ZZ**. But, if you can find enough year-old files to bring the occupancy percentage down to 50%, do that also.

```
RULE 'year-old' MIGRATE FROM POOL 'pool_X'
   THRESHOLD(90,50) WEIGHT(weight_expression)
   TO POOL 'pool_ZZ'
   WHERE  DAYS(CURRENT_TIMESTAMP) — DAYS(ACCESS_TIME) > 365

RULE '3month-old' MIGRATE FROM POOL 'pool_X'
   THRESHOLD(90,80) WEIGHT(weight_expression)
   TO POOL 'pool_ZZ'
   WHERE  DAYS(CURRENT_TIMESTAMP) — DAYS(ACCESS_TIME) > 90
```

More information about weights is available in the next example.

A goal of this **mmapplypolicy** job is to reduce the occupancy percentage of the **FROM POOL** to the low occupancy percentage specified on the **THRESHOLD** clause, if possible. The **mmapplypolicy** job does not migrate or delete more files than are necessary to produce this occupancy percentage. The task consists of these steps:

a. Each candidate file is assigned a weight.

b. All candidate files are sorted by weight.

c. The highest weight files are chosen to **MIGRATE** or **DELETE** until the low occupancy percentage is achieved, or there are no more candidates.

The administrator who writes the rules must ensure that the computed weights are as intended, and that the comparisons are meaningful. This is similar to the IBM Spectrum Protect convention, where the weighting function for each file is determined by the equation:

```
X * access_age + Y * file_size
```

where:

access_age is **DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)**

file_size is **FILE_SIZE** or **KB_ALLOCATED**

X and Y are weight factors that are chosen by the system administrator.

15. The **WEIGHT** clause can be used to express ideas like this (stated informally):

```
IF access_age > 365 days THEN weight = 100000 + access_age
ELSE IF access_age < 30 days THEN weight = 0
ELSE weight= KB_ALLOCATED
```

This rule means:

- Give a very large weight bias to any file older than a year.
- Force the weight of any file younger than 30 days to 0.
- Assign weights to all other files according to the number of kilobytes occupied.

The following codeblock shows the formal SQL syntax:

```
CASE
 WHEN DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 365
   THEN 100000 + DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)
 WHEN DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) < 30
   THEN 0
 ELSE
   KB_ALLOCATED
END
```

16. The **SHOW** clause has no effect in matching files but can be used to define additional attributes to be exported with the candidate file lists. It can be used for any purpose but is primarily used to support file aggregation.

    To support aggregation, you can use the **SHOW** clause to output an aggregation value for each file that is selected by a rule. You can then output those values to a file list and input that list to an external program that groups the files into aggregates.

17. If you have a large number of filesets against which to test, use the **FILESET_NAME** variable as shown in the following example:

    ```
    RULE 'x' SET POOL 'gold' WHERE FILESET_NAME LIKE 'xyz.%.xyz'
    ```

    However, if you are testing against just a few filesets, you can use the FOR FILESET('xyz1', 'xyz2') form instead.

18. You can convert a time interval value to a number of seconds with the SQL cast syntax, as in the following example:

    ```
    define([toSeconds],[(($1) SECONDS(12,6))])

    define([toUnixSeconds],[toSeconds($1 - '1970-1-1@0:00')])

    RULE external list b
    RULE list b SHOW('sinceNow=' toSeconds(current_timestamp-modification_time) )
    RULE external list c
    RULE list c SHOW('sinceUnixEpoch=' toUnixSeconds(modification_time) )
    ```

    The following method is also supported:

```
define(access_age_in_days,( INTEGER(( (CURRENT_TIMESTAMP - ACCESS_TIME) SECONDS)) /(24*3600.0) ) )

RULE external list w exec ''
RULE list w weight(access_age_in_days) show(access_age_in_days)
```

19. You can create a policy that lists the files that are created, accessed, or modified later than a specified timestamp. The timestamp must be converted from native format to UTC format. The following example policy lists all the files that were created after the timestamp 2017-02-21 04:56 IST:

    ```
    cat policy
    RULE 'filesRule' LIST 'files'
         SHOW(varchar(kb_allocated)  || ' ' || varchar(file_size))
         WHERE (CREATION_TIME > TIMESTAMP('LAST_CREATE'))
    ```

    To implement this policy, enter the following commands. The third line converts the time stamp to UTC format.

    ```
    $LC='2017-02-21 04:56 IST'
    echo $LC
    $LCU=$(date +%Y-%m-%d" "%H:%M  -d "$LC" -u)
    echo $LCU
    $mmapplypolicy gpfs0 -P policy -I defer -f /tmp -M LAST_CREATE="$LCU"
    ```

The `/tmp/list.files` file contains the list of selected files.

You can modify the SHOW clause to list any file attribute. For more information, see "File attributes in SQL expressions" on page 371.

Using macro processing utilities with policy rules

Prior to evaluating the policy rules, GPFS invokes the **m4** macro processor to process the policy file.

This processing allows you to incorporate into the policy file some of the traditional **m4** facilities and to define simple and parameterized macros, conditionally include text, perform conditional evaluation, perform simple string operations, perform simple integer arithmetic and much more.

Note: GPFS uses the **m4** built-in **changequote** macro to change the quote pair to [ ] and the **changecom** macro to change the comment pair to /* */ (as in the C programming language).

Utilizing **m4** as a front-end processor simplifies writing policies and produces policies that are easier to understand and maintain. Here is Example 15 on page 388 from "Policy rules: Examples and tips" on page 386 written with a few **m4** style macro definitions:

```
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))

define(weight_expression,
      CASE
       WHEN access_age > 365
         THEN 100000 + access_age
       WHEN access_age < 30
         THEN 0
       ELSE
         KB_ALLOCATED
       END
)

RULE year-old MIGRATE FROM POOL pool_X
   THRESHOLD(90,50) WEIGHT(weight_expression)
   TO POOL pool_ZZ
   WHERE access_age > 365

RULE 3month-old MIGRATE FROM POOL pool_X
   THRESHOLD(90,80) WEIGHT(weight_expression)
   TO POOL pool_ZZ
   WHERE access_age > 90
```

If you would like to use megabytes or gigabytes instead of kilobytes to represent file sizes, and **SUNDAY**, **MONDAY**, and so forth instead of 1, 2, and so forth to represent the days of the week, you can use macros and rules like this:

```
define(MB_ALLOCATED,(KB_ALLOCATED/1024.0))
define(GB_ALLOCATED,(KB_ALLOCATED/1048576.0))
define(SATURDAY,7)
define(SUNDAY,1)
define(MONDAY,2)
define(DAY_OF_WEEK, DayOfWeek(CURRENT_DATE))

RULE 'gb1' WHEN(DAY_OF_WEEK IN (SATURDAY,SUNDAY))
       MIGRATE TO POOL 'ypooly' WHERE GB_ALLOCATED >= .015

RULE 'mb4' MIGRATE TO POOL 'zpoolz' WHERE MB_ALLOCATED >= 4
```

The **mmapplypolicy** command provides a **-M** option that can be used to specify **m4** macro definitions when the command is invoked. The policy rules may include variable identifiers whose values can be set using one or more **-M** options on the **mmapplypolicy** command. The policy rules could then compare file attributes to the currently provided values for the macro defined variables.

Among other things, this allows you to create a single policy file and reuse it for incremental backups without editing the file for each backup. For example, if your policy file contains the rules:

```
RULE EXTERNAL POOL 'archive' EXEC '/opts/hpss/archiveScript' OPTS '-server archive_server'
RULE 'mig1' MIGRATE TO POOL 'dead' WHERE ACCESS_TIME < TIMESTAMP(deadline)
RULE 'bak1' MIGRATE TO POOL 'archive' WHERE MODIFICATION_SNAPID > last_snapid
```

Then, if you invoke **mmapplypolicy** with these options

```
mmapplypolicy ... -M "deadline='2006-11-30'" -M "last_snapid=SNAPID('2006_DEC')" \
-M archive_server="archive.abc.com"
```

The "mig1" rule will migrate old files that were not accessed since 2006/11/30 to an online pool named "dead". The "bak1" rule will migrate files that have changed since the 2006_DEC snapshot to an external pool named "archive". When the external script /opts/hpss/archiveScript is invoked, its arguments will include "-server archive.abc.com".

Managing policies

Policies and the rules that they contain are used to assign files to specific storage pools.

A storage pool typically contains a set of volumes that provide a specific quality of service for a specific use, such as to store all files for a particular application or a specific business division.

Managing policies includes:
- "Creating a policy"
- "Installing a policy" on page 392
- "Changing the active policy" on page 392
- "Listing policies" on page 393
- "Validating policies" on page 393
- "Deleting policies" on page 393

Creating a policy

Create a text file for your policy by following these guidelines.
- A policy must contain at least one rule.
- A policy file is limited to a size of 1 MB.
- The last placement rule of a policy rule list should be of this form, so that if no other placement rules apply to a file, the file will be assigned to a default pool:

```
RULE 'DEFAULT' SET POOL 'default-data-pool'
```

If you do not do this, and no other rule applies, an **EINVAL** error code is returned.

For file systems that are upgraded to V4.1.1 or later: If there are no **SET POOL** policy rules installed to a file system by **mmchpolicy**, the system acts as if the single rule **SET POOL** *'first-data-pool'* is in effect, where *first-data-pool* is the firstmost non-system pool that is available for file data storage, if such a non-system pool is available. ("Firstmost" is the first according to an internal index of all pools.) However, if there are no policy rules installed and there is no non-system pool, the system acts as if **SET POOL 'system'** is in effect.

For file systems that are upgraded to V4.1.1: Until a file system is upgraded, if no **SET POOL** rules are present (set by **mmchpolicy**) for the file system, all data is stored in the **'system'** pool.
- Comments within a policy must start with a **/*** and end with a ***/**:

```
/* This is a comment */
```

For more information, see the topic "Policy rules" on page 363.

Installing a policy

Install a policy by following these guidelines.

To install a policy:

1. Create a text file containing the desired policy rules.
2. Issue the **mmchpolicy** command.

Using thresholds to migrate data between pools

Exhausting space in any one online storage pool generates a NO_SPACE event even though there might be space available in other online storage pools. To create free space, file data can be moved to other online storage pools, deleted, or moved to external storage pools.

Tiered storage solutions can avoid **NO_SPACE** events by monitoring file system space usage and migrating data to other storage pools when the system exceeds a specified threshold. Policies can be used to monitor space usage using a threshold by using the **THRESHOLD** keyword in the policy along with a high water mark and a low water mark. A threshold policy can be applied to a file system by using the `mmchpolicy` command.

A NO_SPACE event is generated if the file system is out of space. A **lowDiskSpace** event requires a threshold. If a threshold is specified, a **lowDiskSpace** event is generated.

GPFS provides user exits for NO_SPACE and **lowDiskSpace** events. Using the **mmaddcallback** command, you can specify a script that runs when either of these events occurs. For more information, see the topic *mmaddcallback command* in the *IBM Spectrum Scale: Command and Programming Reference*.

The file with the policy rules used by **mmapplypolicy** is the one that is currently installed in the file system. It is a good idea for the HSM user to define migration or deletion rules to reduce the usage in each online storage pool. Migration rules that are defined with a high and low THRESHOLD establish the threshold that is used to signal the **lowDiskSpace** event for that pool. Because more than one migration rule can be defined, the threshold for a pool is the minimum of the high thresholds set by the rules for that pool. Each pool has its own threshold. Pools without migration rules do not signal a **lowDiskSpace** event.

In order to enable the low space events required for the policy to work, the **enableLowspaceEvents** global parameter must be set to 'yes'. To view the current status of this setting, run `mmlsconfig` and **mmchconfig enableLowspaceEvents=yes** to set it if necessary.

Note: GPFS must be restarted on all nodes in order for this setting to take effect.

A callback must be added in order to trigger the policy run when the low space event is generated. A simple way to add the callback is using the `mmstartpolicy` command.

To add a callback, run this command. The following command is on one line:

```
mmaddcallback MIGRATION --command /usr/lpp/mmfs/bin/mmstartpolicy --event lowDiskSpace
--parms "%eventName %fsName --single-instance
```

The **--single-instance** flag is required to avoid running multiple migrations on the file system at the same time.

Changing the active policy

When you prepare a file with the new or changed policy rules, then issue the **mmchpolicy** command.

The **mmchpolicy** command activates the following sequence of events:

1. The policy file is read into memory, and the information is passed to the current file system manager node.
2. The policy rules are validated by the file system manager.

3. If the policy file contains incorrect rules, no updates are made and an error is returned.
4. If no errors are detected, the new policy rules are installed in an internal file.

Policy changes take effect immediately on all nodes that have the affected file system mounted. For nodes that do not have the file system mounted, policy changes take effect upon the next mount of the file system.

Listing policies

When you use the **mmlspolicy** command to list policies, follow these guidelines.

The **mmlspolicy** command displays policy information for a given file system. The information displayed is:
- When the policy file was installed.
- The user who installed the policy file.
- The first line of the original policy file.

The **mmlspolicy -L** command returns the installed (original) policy file. This shows all the rules and comments as they were in the policy file when it was installed. This is useful if you want to change policy rules - simply retrieve the original policy file using the **mmlspolicy -L** command and edit it.

Validating policies

When you validate a policy file, follow this guideline.

The **mmchpolicy -I test** command validates but does *not* install a policy file.

Deleting policies

When you remove the current policy rules and restore the file-placement policy, follow this guideline.

To remove the current policy rules and restore the default GPFS file-placement policy, specify **DEFAULT** as the name of the policy file on the **mmchpolicy** command. This is equivalent to installing a policy file with just one rule:

```
RULE 'DEFAULT' SET POOL 'system'
```

Improving performance with the --sort-command parameter

To improve performance of the **mmapplypolicy** command, follow these guidelines.

One possible way to improve the performance of the **mmapplypolicy** command is to specify an alternative sort command to be used instead of the default sort command provided by the operating system. To do this, issue **mmapplypolicy --sort-command** *SortCommand*, specifying the executable path of the alternative command.

For example, on AIX the GNU **sort** program, freely available within the `coreutils` package from AIX Toolbox for Linux Applications (www.ibm.com/systems/power/software/aix/linux/toolbox), will typically perform large sorting tasks much faster than the standard AIX sort command. If you wanted to specify the GNU sort program, you would use the following command: `mmapplypolicy --sort-command /opt/freeware/bin/sort`.

Before issuing **mmapplypolicy --sort-command** on a large number of files, first do a performance comparison between the alternative sort command and the default sort command on a smaller number of files to determine whether the alternative command is in fact faster.

If you specify an alternative sort command, it is recommended that you install it on all cluster nodes.

Improvements in performance in very large file systems

Read about how to improve the performance of the `mmapplypolicy` command in very large file systems.

The following actions can improve performance:
- Put the `system` pool on the fastest storage available, which can be either solid-state storage or hard disks. In either case, spread the system pool over multiple storage devices, so that they can seek in parallel, independently of one another. Verify that logical disks do not map to the same physical disk.

  The policy engine requires many I/O operations against file system metadata. Storing metadata on the fastest storage possible can improve the performance of policy execution.
- Include both the `-N` parameter and the `-g` parameter on the `mmapplypolicy` command line.
  - The `-N` parameter specifies a list of nodes that run parallel instances of policy code.
  - The `-g` parameter specifies a global work directory that can be accessed by the nodes that are specified by the `-N` parameter.
  - When both `-N` and `-g` are specified, `mmapplypolicy` uses high-performance and fault-tolerant protocols during execution.
- If the exact order in which files are processed is not important, consider specifying the `--choice-algorithm fast` algorithm, which works with the `-N` and `-g` options for parallel processing.
- If the order in which files are processed is not important at all, specify `WEIGHT(0)` in your `MIGRATE`, `LIST`, and `DELETE` policy rules.
- Update the file system format to format level 13.01 (GPFS 3.5.0.1) or higher. File systems at this level can support the following two features, among others:
  - Storing small directories and small files in the inode.
  - Fast extended attributes. For this feature, you must also update the file system by running `mmmigratefs`.

  These two features can improve the performance of the `mmapplypolicy` command. See the following links:

  Chapter 14, "File system format changes between versions of IBM Spectrum Scale," on page 153

  Completing the migration to a new level of IBM Spectrum Scale in the *IBM Spectrum Scale: Administration Guide*

Working with external storage pools

With external storage pools you can migrate files to storage pools managed by an external application such as IBM Spectrum Protect.

The following topics describe how to work with external storage pools:
- Defining the external pools
- "User-provided program for managing external pools" on page 395
- "File list format" on page 396
- "Record format" on page 396
- "Migrate and recall with external pools" on page 397
- "Pre-migrating files with external storage pools" on page 398
- "Purging files from external storage pools" on page 399
- "Using thresholds to migrate data between pools" on page 392

Defining external pools

When you define external pools, follow these rules.

GPFS file management policy rules control data migration into external storage pools. Before you can write a migration policy you must define the external storage pool that the policy will reference. After you define the storage pool, you can then create policies that set thresholds that trigger data migration into or out of the referenced external pool.

When a storage pool reaches the defined threshold or when you invoke **mmapplypolicy**, GPFS processes the metadata, generates a list of files, and invokes a user provided script or program which initiates the appropriate commands for the external data management application to process the files. This allows GPFS to transparently control offline storage and provide a tiered storage solution that includes tape or other media.

Before you can migrate data to an external storage pool, you must define that pool. To define external storage pools, use a GPFS policy rule as follows:

```
RULE EXTERNAL POOL 'PoolName' EXEC 'InterfaceScript' [OPTS 'OptionsString'] [ESCAPE 'SpecialCharacters']
```

Where:
- *PoolName* defines the name of the storage pool
- *InterfaceScript* defines the program or script to be invoked to migrate data to or from the external pool
- *OptionsString* is an optional string that, if provided, will be passed to the *InterfaceScript*

You must have a separate EXTERNAL POOL rule for each external pool that you wish to define.

Example of a rule that defines a storage pool

The following rule defines a storage pool called `externalpoolA`.

```
RULE EXTERNAL POOL 'externalpoolA' EXEC '/usr/hsm/bin/hsmControl' OPTS '-server=hsm-manager.nyc.com'
```

In this example:
- `externalpoolA` is the name of the external pool
- `/usr/hsm/bin/hsmControl` is the location of the executable script that will be invoked when there are files for migration
- `-server=hsm-manager.nyc.com` is the location of storage pool `externalpoolA`

For additional information, refer to "User-provided program for managing external pools."

User-provided program for managing external pools

After you define an external storage pool, subsequent migration or deletion rules might refer to that pool as a source or target storage pool.

When the **mmapplypolicy** command is invoked and a rule dictates that data should be moved to or from an external pool, the user provided program identified with the **EXEC** clause in the policy rule launches. That executable program receives three arguments:
- The command to be executed. Your script should implement each of the following sub-commands:
  - LIST - Provides arbitrary lists of files with no semantics on the operation.
  - MIGRATE - Migrate files to external storage and reclaim the online space allocated to the file.
  - PREMIGRATE - Migrate files to external storage but do not reclaim the online space.
  - PURGE - Delete files from both the online file system and the external storage.
  - RECALL - Recall files from external storage to the online storage.
  - TEST – Test for presence and operation readiness. Return zero for success. Return non-zero if the script should not be used on a given node.
- The name of a file containing a list of files to be migrated, premigrated, or purged. See "File list format" on page 396 for detailed description of the layout of the file.

- Any optional parameters specified with the OPTS clause in the rule. These optional parameters are not interpreted by the GPFS policy engine.

The **mmapplypolicy** command invokes the external pool script on all nodes in the cluster that have installed the script in its designated location. The script must be installed at the node that runs **mmapplypolicy**. You can also install the script at other nodes for parallel operation but that is not required. GPFS may call your exit script one or more times for each command.

Important: Use the **EXCLUDE** rule to exclude any special files that are created by an external application. For example, when using IBM Spectrum Protect or Hierarchical Storage Management (HSM), exclude the **.SpaceMan** directory to avoid migration of **.SpaceMan**, which is an HSM repository.

File list format

Each call to the external pool script specifies the pathname for a temporary file that contains a list of files to be operated on.

This file list defines one file per line as follows:

```
InodeNumber GenNumber SnapId [OptionalShowArgs] -- FullPathToFile
```

where:
- *InodeNumber* is a 64-bit inode number.
- *GenNumber* is a 32-bit file generation number.
- *SnapId* is a 64-bit snapshot identifier.
- *OptionalShowArgs* is the result, if any, from the evaluation of the SHOW clause in the policy rule.
- *FullPathToFile* is a fully qualified path name to the file. When there are multiple paths within a file system to a particular file (*Inode*, *GenNumber*, and *SnapId*), each path is shown.
- The "--" characters are a field delimiter that separates the optional show parameters from the path name to the file.

Note: GPFS does not restrict the character set used for path and file names. All characters except '\0' are valid. To make the files readily parseable, files or directories containing the newline character and/or other special characters are "escaped", as described previously, in connection with the ESCAPE `'%special-characters'` clause.

Record format

The format of the records in each file list file can be expressed as shown in the following example.

Each file list file:

```
iAggregate:WEIGHT:INODE:GENERATION:SIZE:iRule:resourceID:attr_flags:
path-length!PATH_NAME:pool-length!POOL_NAME
[;show-length>!SHOW]end-of-record-character
```

where:
- *iAggregate* is a grouping index that is assigned by **mmapplypolicy**.
- *WEIGHT* represents the **WEIGHT** policy language file attribute.
- *INODE* represents the **INODE** policy language file attribute.
- *GENERATION* represents the **GENERATION** policy language file attribute.
- *SIZE* represents the **SIZE** policy language file attribute.
- *iRule* is a rule index number assigned by **mmapplypolicy**, which relates to the policy rules file that is supplied with the **-P** argument.
- *resourceID* represents a pool index, **USER_ID**, **GROUP_ID**, or fileset identifier, depending on whether thresholding is done with respect to pool usage or to user, group, or fileset quotas.

- *attr_flags* represents a hexadecimal encoding of some of the attributes that are also encoded by the policy language variable *MISC_ATTRIBUTES*. The low-order 20 bits of *attr_flags* are taken from the **ia_flags** word that is defined in the **gpfs.h** API definition.
- *path-length* represents the length of the character string PATH_NAME.
- *pool-length* represents the length of the character string POOL_NAME.
- *show-length* represents the length of the character string SHOW.
- *end-of-record-character* is **\n** or **\0**.

Note: You can only change the values of the *iAggregate*, *WEIGHT*, *SIZE*, and *attr_flags* fields. Changing the values of other fields can cause unpredictable policy execution results.

All of the numeric fields are represented as hexadecimal strings, except the *path-length*, *pool-length*, and *show-length* fields, which are decimal encoded. These fields can be preceded by a minus sign (**-**), which indicates that the string that follows it contains escape sequences. In this case, the string might contain occurrences of the character pair \n, which represents a single newline character with a hexadecimal value of 0xA in the filename. Also, the string might contain occurrences of the character pair \\, which represents a single \ character in the filename. A \ will only be represented by \\ if there are also newline characters in the filename. The value of the length field within the record counts any escape characters.

The encoding of *WEIGHT* is based on the 64-bit IEEE floating format, but its bits are *flipped* so that when a file list is sorted using a conventional collating sequence, the files appear in decreasing order, according to their *WEIGHT*.

The encoding of *WEIGHT* can be expressed and printed using C++ as:

```
double w =  - WEIGHT;
/* This code works correctly on big-endian and little-endian systems */
uint64 u = *(uint64*)&w;  /* u is a 64 bit long unsigned integer
     containing the IEEE 64 bit encoding of the double floating point
     value of variable w */
     uint64 hibit64 = ((uint64)1<<63);
if (w < 0.0) u = ~u; /* flip all bits */
else u = u | hibit64; /* force the high bit from 0 to 1,
     also handles both "negatively" and "positively" signed 0.0 */
printf("%016llx",u);
```

The format of the majority of each record can be expressed in C++ as:

```
printf("%03x:%016llx:%016llx:%llx:%llx:%x:%x:%llx:%d!%s:%d!%s",
  iAggregate, u /*encoding of −1*WEIGHT from above*/, INODE, ... );
```

Notice that the first three fields are fixed in length to facilitate the sorting of the records by the field values *iAggregate*, *WEIGHT*, and *INODE*.

The format of the optional SHOW string portion of the record can be expressed as:

```
if(SHOW && SHOW[0]) printf(";%d!%s",strlen(SHOW),SHOW);
```

For more information, see the topic *mmapplypolicy command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Migrate and recall with external pools

After you define an external storage pool, subsequent migration or deletion rules might refer to that pool as a source or target storage pool.

When you invoke **mmapplypolicy** and a rule dictates that data should be deleted or moved to or from an external pool, the program identified in the EXTERNAL POOL rule is invoked with the following arguments:

- The command to be executed.
- The name of the file containing a list of files to be migrated, pre-migrated, or purged.
- Optional parameters, if any.

For example, let us assume an external pool definition:
```
RULE EXTERNAL POOL 'externalpoolA'
    EXEC '/usr/hsm/bin/hsmControl' OPTS '-server=hsm-manager.nyc.com'
```

To move files from the internal **system** pool to storage pool "externalpoolA" you would simply define a migration rule that may look something like this:
```
RULE 'MigToExt' MIGRATE FROM POOL('system') TO POOL('externalpoolA') WHERE ...
```

This would result in the external pool script being invoked as follows:
```
/usr/hsm/bin/hsmControl MIGRATE /tmp/filelist -server=hsm-manager.nyc.com
```

Similarly, a rule to migrate data from an external pool back to an internal storage pool could look like:
```
RULE 'MigFromExt' MIGRATE FROM POOL 'externalpoolA' TO POOL 'system' WHERE ...
```

This would result in the external pool script being invoked as follows:
```
/usr/hsm/bin/hsmControl RECALL /tmp/filelist -server=hsm-manager.nyc.com
```

Notes:
1. When migrating to an external storage pool, GPFS ignores the LIMIT and REPLICATION clauses in the policy rule.
2. If you are using HSM with external storage pools, you may need to create specific rules to avoid system problems. These rules should exclude HSM-related system files from both migration and deletion. These rules use the form:
   ```
   RULE 'exclude hsm system files' EXCLUDE WHERE PATH_NAME LIKE '%/.SpaceMan%'
   ```

Pre-migrating files with external storage pools
Pre-migration is a standard technique of Hierarchical Storage Management (HSM) systems such as IBM Spectrum Protect.

Pre-migration copies data from GPFS internal storage pools to external pools but leaves the original data online in the active file system. Pre-migrated files are often referred to as "dual resident" to indicate that the data for the files are available both online in GPFS and offline in the external storage manager. Files in the pre-migrated state allow the external storage manager to respond more quickly to low space conditions by simply deleting the copy of the file data that is stored online.

The files to be pre-migrated are determined by the policy rules that migrate data to an external storage pool. The rule will select files to be migrated and optionally select additional files to be pre-migrated. The THRESHOLD clause of the rule determines the files that need to be pre-migrated.

If you specify the THRESHOLD clause in file migration rules, the **mmapplypolicy** command selects files for migration when the affected storage pool reaches the specified high occupancy percentage threshold. Files are migrated until the storage pool utilization is reduced to the specified low occupancy percentage threshold. When migrating to an external storage pool, GPFS allows you to specify a third pool occupancy percentage which defines the file pre-migration threshold: after the low occupancy percentage is reached, files are pre-migrated until the pre-migration occupancy percentage is reached.

To explain thresholds in another way, think of an internal storage pool with a high threshold of 90%, a low threshold of 80%, and a pre-migrate threshold of 60%. When this internal storage pool reaches 90% occupancy, the policy rule will migrate files until the occupancy of the pool reaches 80% then it will continue to pre-migrate another 20% of the file space until the 60% threshold is reached.

Pre-migration can only be done with external storage managers using the XDSM Data Storage Management API (DMAPI). Files in the migrated and pre-migrated state will have a DMAPI managed region set on the file data. Files with a managed region are visible to **mmapplypolicy** and may be referenced by a policy rule. You can approximate the amount of pre-migrated space required by counting the space used after the end of the first full data block on all files with managed regions.

Note:

1. If you do not set a pre-migrate threshold or if you set a value that is greater than or equal to the low threshold, then GPFS will not pre-migrate files. This is the default setting.
2. If you set the pre-migrate threshold to zero, then GPFS will pre-migrate all files.

Purging files from external storage pools

Files that have been migrated to an external storage pool continue to have their file name and attributes stored in GPFS; only the file data has been migrated. Files that have been migrated or pre-migrated to an external storage pool may be deleted from the GPFS internal storage pool and from the external storage pool with the policy language using a DELETE rule.

```
RULE 'DelFromExt' DELETE WHERE ...
```

If the file has been migrated or pre-migrated, this would result in the external pool script being invoked as follows:

```
/usr/hsm/bin/hsmControl PURGE /tmp/filelist -server=hsm-manager.nyc.com
```

The script should delete a file from both the online file system and the external storage manager. However, most HSM systems automatically delete a file from the external storage manager whenever the online file is deleted. If that is how your HSM system functions, your script will only have to delete the online file.

Backup and restore with storage pools

When you back up data or restore data to a storage pool, consider the following descriptions.

You can use the GPFS ILM tools to backup data for disaster recovery or data archival to an external storage manager such as the IBM Spectrum Protect Backup-Archive client. When backing up data, the external storage manager must preserve the file name, attributes, extended attributes, and the file data. Among other things, the extended attributes of the file also contain information about the assigned storage pool for the file. When you restore the file, this information is used to assign the storage pool for the file data.

The file data may be restored to the storage pool to which it was assigned when it was backed up or it may be restored to a pool selected by a restore or placement rule using the backed up attributes for the file. GPFS supplies three subroutines that support backup and restore functions with external pools:

- **gpfs_fgetattrs()**
- **gpfs_fputattrs()**
- **gpfs_fputattrswithpathname()**

GPFS exports the extended attributes for a file, including its ACLs, using **gpfs_fgetattrs()**. Included in the extended attributes is the name of the storage pool to which the file has been assigned, as well as file attributes that are used for file placement. When the file is restored the extended attributes are restored using either **gpfs_fputattrs()** or **gpfs_fputattrswithpathname()**.

When a backup application uses **gpfs_fputattrs()** to restore the file, GPFS assigns the restored file to the storage pool with the same name as when the file was backed up. Thus by default, restored files are assigned to the same storage pool they were in when they were backed up. If that pool is not available, GPFS tries to select a pool using the current file placement rules. If that fails, GPFS assigns the file to the system storage pool.

Note: If a backup application uses **gpfs_fputattrs()** to restore a file, it will omit the **RESTORE** RULE.

When a backup application restores the file using **gpfs_fputattrswithpathname()**, GPFS is able to access additional file attributes that may have been used by placement or migration policy rules to select the storage pool for the file. This information includes the UID and GID for the owner, the access time for the file, file modification time, file size, the amount of storage allocated, and the full path to the file. GPFS uses **gpfs_fputattrswithpathname()** to match this information with restore policy rules you define.

In other words, the **RESTORE** rule looks at saved file attributes rather than the current file attributes. The call to **gpfs_fputattrswithpathname()** tries to match the saved information to a **RESTORE** rule. If the **RESTORE** rules cannot match saved attributes, GPFS tries to restore the file to the same storage pool it was in when the file was backed up. If that pool is not available GPFS tries to select a pool by matching placement rules. If that fails, GPFS assigns the file to the system storage pool.

Note: When a **RESTORE** rule is used, and restoring the file to the specified pool would exceed the occupancy percentage defined for that pool, GPFS skips that rule and the policy engine looks for the next rule that matches. While testing for matching rules, GPFS takes into account the specified replication factor and the **KB_ALLOCATED** attribute of the file that is being restored.

The **gpfs_fgetattrs()**, **gpfs_fputattrs()**, and **gpfs_fputattrswithpathname()** subroutines have optional flags that further control the selection of storage pools. For more information, see the topics *gpfs_fgetattrs() subroutine*, *gpfs_fputattrs() subroutine*, and *gpfs_fputattrswithpathname() subroutine* in the *IBM Spectrum Scale: Command and Programming Reference*.

Working with external lists

External lists, like external pools, generate lists of files. For external pools, the operations on the files correspond to the rule that references the external pool. For external lists, there is no implied operation; it is simply a list of files that match the criteria specified in the policy rule.

External lists must be defined before they can be used. External lists are defined by:

```
RULE EXTERNAL LIST 'ListName' EXEC 'InterfaceScript' [OPTS 'OptionsString'] [ESCAPE 'SpecialCharacters']
```

Where:
- *ListName* defines the name of the external list
- *InterfaceScript* defines the program to be invoked to operate on the list of files
- *OptionsString* is an optional string that, if provided, will be passed to the *InterfaceScript*

See "User-provided program for managing external pools" on page 395.

Example

The following rule defines an external list called `listfiles`:

```
RULE EXTERNAL LIST 'listfiles' EXEC '/var/mmfs/etc/listControl' OPTS '-verbose'
```

In this example:
- `listfiles` is the name of the external list
- `/var/mmfs/etc/listControl` is the location of the executable script that defines the operations on the list of files
- `-verbose` is an optional flag to the `listControl` script

The EXTERNAL LIST rule provides the binding between the lists generated with regular LIST rules and the external program that you want to run with these lists as input. For example, this rule would generate a list of all files that have more than 1 MB of data in an internal storage pool:

```
RULE 'ListLargeFiles' LIST 'listfiles' WHERE KB_ALLOCATED > 1024
```

By default, only user files are included in lists. To include directories, symbolic links, and other file system objects, the DIRECTORIES_PLUS clause must be specified. For example, this rule would generate a list of all objects in the file system.

```
RULE 'ListAllObjects' LIST 'listfiles' DIRECTORIES_PLUS
```

ILM for snapshots

ILM for snapshots can be used to migrate snapshot data.

Similar to the files in the root file system, snapshot data can also be managed by using policy rules. Rules can be written to migrate snapshot data among internal storage pools or generated in specific pools.

Snapshot data migration

Snapshot data can be migrated by using the **mmapplypolicy** command with simple migration rules. For example, to migrate data of a snapshot with the name snapname from an SSD pool to the Capacity pool, use the following rule:

```
RULE 'MigToCap' MIGRATE FROM POOL 'SSD' TO POOL 'Capacity'
```

Then, run the **mmapplypolicy** command with the **-S snapname** parameter to complete the migration.

Snapshot data belonging to AFM and AFM DR can also be migrated. Use the following rule:

```
RULE 'migrate' MIGRATE FROM POOL 'POOL1' TO POOL 'POOL2'
```

In this example, data is migrated from POOL1 to POOL2. You must exclude files which are internal to AFM while migrating snapshot data. An example of a rule to exclude such files is as under:

```
RULE 'migrate' MIGRATE FROM POOL 'POOL1' TO POOL 'POOL2' WHERE
( NOT (PATH_NAME LIKE '/%/.afm%') OR (PATH_NAME LIKE '/%/.ptrash%')
OR (PATH_NAME LIKE '/%/.afmtrash%')OR (PATH_NAME LIKE '/%/.pconflicts%'))
```

Note:

- The snapshot data cannot be migrated to external pools.
- The migration rules for snapshot data cannot be mixed with other rule types.
- SetXattr file function is not allowed on both the MIGRATE and SET SNAP_POOL rules for snapshot files.

Snapshot data placement

A snapshot placement rule can be used to generate snapshot data in specific internal pools. For example, to generate the snapshot data for all snapshots in the Capacity pool, use the following rule:

```
RULE 'SnapPlacement' SET SNAP_POOL 'Capacity'
```

Snapshot data for specific snapshots can be placed in specific pools by using the following rule:

```
RULE 'SnapPlacement' SET SNAP_POOL 'Capacity' WHERE SNAP_NAME LIKE '%daily%'
```

Include this rule in the set of rules installed for the file system. Placement of a snapshot file happens when the first data block is copied to it because of the changes made to the file in the root file system.

Note: Snapshot data cannot be placed in external pools.

The placement rule can be applied to snapshot data belonging to AFM and AFM DR. In the following example, snap pool is set as POOL1, for all snapshots having psnap as a sub-pattern in the name.

```
RULE 'setsnappool' SET SNAP_POOL 'POOL1' WHERE SNAP_NAME LIKE '%psnap%'
```

Another example is as under -

```
RULE 'setsnappool' SET SNAP_POOL 'POOL1' WHERE SNAP_NAME LIKE '%afm%'
```

Ill placement of snapshot files

Deletion of files can result in these files moving to snapshot and then becoming ill-placed or ill-replicated. In these cases, the `mmrestripefile` command can be used to correct the ill placement and ill replication of snapshot files.

Filesets

In most file systems, a file hierarchy is represented as a series of directories that form a tree-like structure. Each directory contains other directories, files, or other file-system objects such as symbolic links and hard links. Every file system object has a name associated with it, and is represented in the namespace as a node of the tree.

In addition, GPFS utilizes a file system object called a *fileset*. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. Filesets provide a means of partitioning the file system to allow administrative operations at a finer granularity than the entire file system:

* Filesets can be used to define quotas on both data blocks and inodes.
* The owning fileset is an attribute of each file and can be specified in a policy to control initial data placement, migration, and replication of the file's data. See "Policies for automating file management" on page 362.
* Fileset snapshots can be created instead of creating a snapshot of an entire file system.

GPFS supports independent and dependent filesets. An independent fileset is a fileset with its own inode space. An inode space is a collection of inode number ranges reserved for an independent fileset. An inode space enables more efficient per-fileset functions, such as fileset snapshots. A dependent fileset shares the inode space of an existing, independent fileset. Files created in a dependent fileset are assigned inodes in the same collection of inode number ranges that were reserved for the independent fileset from which it was created.

When the file system is created, only one fileset, called the *root* fileset, exists. The root fileset is an independent fileset that cannot be deleted. It contains the root directory as well as any system files such as quota files. As new files and directories are created, they automatically become part of the parent directory's fileset. The fileset to which a file belongs is largely transparent for ordinary file access, but the containing fileset can be displayed along with the other attributes of each file using the **mmlsattr -L** command.

The root directory of a GPFS file system is also the root of the root fileset.

Fileset namespace

A newly created fileset consists of an empty directory for the root of the fileset, and it is initially not linked into the file system's namespace. A newly created fileset is not visible to the user until it is attached to the namespace by issuing the **mmlinkfileset** command.

Filesets are attached to the namespace with a special link called a *junction*. A junction is a special directory entry, much like a POSIX hard link, that connects a name in a directory of one fileset (source) to the root directory of another fileset (target). A fileset may be the target of only one junction, so that a fileset has a unique position in the namespace and a unique path to any of its directories. The target of the junction is referred to as the *child fileset*, and a fileset can have any number of children. From the user's viewpoint, a junction always appears as if it were a directory, but the user is not allowed to issue the **unlink** or **rmdir** commands on a junction.

Once a fileset has been created and linked into the namespace, an administrator can unlink the fileset from the namespace by issuing the **mmunlinkfileset** command. This makes all files and directories within the fileset inaccessible. If other filesets were linked below it, the other filesets become inaccessible,

but they do remain linked and will become accessible again when the fileset is re-linked. Unlinking a fileset, like unmounting a file system, fails if there are open files. The **mmunlinkfileset** command has a force option to close the files and force the unlink. If there are open files in a fileset and the fileset is unlinked with the force option, future references to those files will result in **ESTALE** errors. Once a fileset is unlinked, it can be re-linked into the namespace at its original location or any other location (it cannot be linked into its children since they are not part of the namespace while the parent fileset is unlinked).

The namespace inside a fileset is restricted to a single, connected subtree. In other words, a fileset has only one root directory and no other entry points such as hard links from directories in other filesets. Filesets are always connected at the root directory and only the junction makes this connection. Consequently, hard links cannot cross fileset boundaries. Symbolic links, of course, can be used to provide shortcuts to any file system object in the namespace.

The root fileset is an exception. The root fileset is attached to the local namespace using the standard **mount** command. It cannot be created, linked, unlinked or deleted using the GPFS fileset commands.

See "Managing filesets" on page 405.

Filesets and quotas

The GPFS quota commands support the **-j** option for fileset block and inode allocation.

The quota limit on blocks and inodes in a fileset are independent of the limits for specific users or groups of users. See the following command in the *IBM Spectrum Scale: Command and Programming Reference*:
* **mmdefedquota**
* **mmdefedquotaon**
* **mmdefedquotaoff**
* **mmedquota**
* **mmlsquota**
* **mmquotaoff**
* **mmquotaon**
* **mmrepquota**

In addition, see the description of the **--perfileset-quota** parameter of the following commands:
* **mmchfs**
* **mmcrfs**
* **mmlsfs**

Filesets and storage pools

Filesets are not specifically related to storage pools, although each file in a fileset physically resides in blocks in a storage pool. This relationship is many-to-many; each file in the fileset can be stored in a different user storage pool.

A storage pool can contain files from many filesets. However, all of the data for a particular file is wholly contained within one storage pool.

Using file-placement policies, you can specify that all files created in a particular fileset are to be stored in a specific storage pool. Using file-management policies, you can define how files in a specific fileset are to be moved or deleted during the file's life cycle. See "Policy rules: Terms" on page 366.

Filesets and global snapshots

A GPFS global snapshot preserves the contents of the entire file system, including all its filesets, even unlinked ones.

The state of filesets in the snapshot is unaffected by changes made to filesets in the active file system, such as unlink, link or delete. The saved file system can be accessed through the **.snapshots** directories and the namespace, including all linked filesets, appears as it did when the snapshot was created. Unlinked filesets are inaccessible in the snapshot, as they were in the active file system. However, restoring a snapshot also restores the unlinked filesets, which can then be re-linked and accessed.

If a fileset is included in a global snapshot, it can be deleted but it is not entirely removed from the file system. In this case, the fileset is emptied of all contents and given a status of 'deleted'. The contents of a fileset remain available in the snapshots that include the fileset (that is, through some path containing a **.snapshots** component) even after the fileset is deleted, since all the contents of the fileset are saved when a snapshot is created. The fileset remains in the deleted state until the last snapshot containing it is deleted, at which time the fileset is automatically deleted.

A fileset is included in a global snapshot if the snapshot is created after the fileset was created. Deleted filesets appear in the output of the **mmlsfileset** and **mmlsfileset --deleted** commands, and the **-L** option can be used to display the latest snapshot that includes a fileset.

During a restore from a global snapshot, attributes of filesets included in the snapshot can be altered. The filesets included in the global snapshot are restored to their former state, and newer filesets are deleted. Also, restore may undelete deleted filesets and change linked filesets to unlinked or vice versa. If the name of a fileset was changed since the snapshot was taken, the old fileset name will be restored.

Fileset-level snapshots

Instead of creating a global snapshot of an entire file system, a fileset snapshot can be created to preserve the contents of a single independent fileset plus all dependent filesets that share the same inode space.

If an independent fileset has dependent filesets that share its inode space, then a snapshot of the independent fileset will also include those dependent filesets. In other words, a fileset snapshot is a snapshot of the whole inode space.

Each independent fileset has its own hidden **.snapshots** directory in the root directory of the fileset that contains any fileset snapshots. The **mmsnapdir** command allows setting an option that makes global snapshots also available through **.snapshots** in the root directory of all independent filesets. The **.snapshots** directory in the file system root directory lists both global snapshots and fileset snapshots of the root fileset (the root fileset is an independent fileset). This behavior can be customized with the **mmsnapdir** command.

Fileset snapshot names need not be unique across different filesets, so it is valid to use the same name for fileset snapshots of two different filesets because they will appear under **.snapshots** in two different fileset root directories.

You can restore independent fileset snapshot data and attribute files with the **mmrestorefs** command. For complete usage information, see the topic *mmrestorefs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Filesets and backup

The **mmbackup** command and IBM Spectrum Protect are unaware of the existence of filesets. When restoring a file system that had been backed up to IBM Spectrum Protect, the files are restored to their original path names, regardless of the filesets of which they were originally a part.

IBM Spectrum Protect has no mechanism to create or link filesets during restore. Therefore, if a file system is migrated to IBM Spectrum Protect and then filesets are unlinked or deleted, restore or recall of the file system does not restore the filesets.

During a full restore from backup, all fileset information is lost and all files are restored into the root fileset. It is recommended that you save the output of the **mmlsfileset** command to aid in the reconstruction of fileset names and junction locations. Saving **mmlsfileset -L** also allows reconstruction of fileset comments. Both command outputs are needed to fully restore the fileset configuration.

A partial restore can also lead to confusion if filesets have been deleted, unlinked, or their junctions moved, since the backup was made. For example, if the backed up data was in a fileset that has since been unlinked, the restore process puts it into files and directories in the parent fileset. The unlinked fileset cannot be re-linked into the same location until the restored data is moved out of the way. Similarly, if the fileset was deleted, restoring its contents does not recreate the deleted fileset, but the contents are instead restored into the parent fileset.

Since the **mmbackup** command operates by traversing the directory structure, it does not include the contents of unlinked filesets, even though they are part of the file system. If it is desired to include these filesets in the backup, they should be re-linked, perhaps into a temporary location. Conversely, temporarily unlinking a fileset is a convenient mechanism to exclude it from a backup.

Note: It is recommended not to unlink filesets when doing backups. Unlinking a fileset during an **mmbackup** run can cause the following:
- failure to back up changes in files that belong to an unlinked fileset
- expiration of files that were backed up in a previous **mmbackup** run

In summary, fileset information should be saved by periodically recording **mmlsfileset** output somewhere in the file system, where it is preserved as part of the backup process. During restore, care should be exercised when changes in the fileset structure have occurred since the backup was created.

Attention: If you are using the IBM Spectrum Protect Backup-Archive client you must use caution when you unlink filesets that contain data backed up by IBM Spectrum Protect. IBM Spectrum Protect tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to IBM Spectrum Protect that you deleted the contents of the fileset. Therefore, the IBM Spectrum Protect Backup-Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

Managing filesets

Managing your filesets includes:
- "Creating a fileset"
- "Deleting a fileset" on page 406
- "Linking a fileset" on page 406
- "Unlinking a fileset" on page 407
- "Changing fileset attributes" on page 407
- "Displaying fileset information" on page 407

Creating a fileset

Filesets are created with the **mmcrfileset** command.

By default, filesets are created as dependent filesets that share the inode space of the root. The **--inode-space** *ExistingFileset* option can be used to create a dependent fileset that shares inode space with an existing fileset. The **--inode-space new** option can be used to create an independent fileset with its own dedicated inode space.

A newly created fileset consists of an empty directory for the root of the fileset and it is initially not linked into the existing namespace. Consequently, a new fileset is not visible and files cannot be added to

it, but the fileset name is valid and the administrator can establish quotas on it or policies for it. The administrator must link the fileset into its desired location in the file system's namespace by issuing the **mmlinkfileset** command in order to make use of it.

After the fileset is linked, the administrator can change the ownership and permissions for the new root directory of the fileset, which default to **root** and 0700, to allow users access to it. Files and directories copied into or created within the directory of the fileset become part of the new fileset.

Note the following restrictions on fileset names:
- The name must be unique within the file system.
- The length of the name must be in the range 1-255.
- The name **root** is reserved for the fileset of the root directory of the file system.
- The name cannot be the reserved word *new*. However, the character string new can appear within a fileset name.
- The name cannot begin with a hyphen (-).
- The name cannot contain the following characters: / ? $ & * ( ) ` # | [ ] \
- The name cannot contain a white-space character such as blank space or tab.

For more information, see the topics *mmcrfileset command* and *mmlinkfileset command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Deleting a fileset
Filesets are deleted with the **mmdelfileset** command.

There are several notes to keep in mind when deleting filesets:
- The root fileset cannot be deleted.
- A fileset that is not empty cannot be deleted unless the **-f** flag is specified.
- A fileset that is currently linked into the namespace cannot be deleted until it is unlinked with the **mmunlinkfileset** command.
- A dependent fileset can be deleted at any time.
- An independent fileset cannot be deleted if it has any dependent filesets or fileset snapshots.
- Deleting a dependent fileset that is included in a fileset or global snapshot removes it from the active file system, but it remains part of the file system in a deleted state.
- Deleting an independent fileset that is included in any global snapshots removes it from the active file system, but it remains part of the file system in a deleted state.
- A fileset in the deleted state is displayed in the **mmlsfileset** output with the fileset name in parenthesis. If the **-L** flag is specified, the latest including snapshot is also displayed. The **--deleted** option of the **mmlsfileset** command can be used to display only deleted filesets.
- The contents of a deleted fileset are still available in the snapshot, through some path name containing a **.snapshots** component, because it was saved when the snapshot was created.
- When the last snapshot that includes the fileset has been deleted, the fileset is fully removed from the file system.

For complete usage information, see the topics *mmdelfileset command*, *mmlsfileset command*, and *mmunlinkfileset command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Linking a fileset
After the fileset is created, a junction must be created to link it to the desired location in the file system's namespace using the **mmlinkfileset** command.

The file system must be mounted in order to link a fileset. An independent fileset can be linked into only one location anywhere in the namespace, specified by the *JunctionPath* parameter:

- The root directory
- Any subdirectory
- The root fileset or to any other fileset

A dependent fileset can only be linked inside its own inode space.

If *JunctionPath* is not specified, the junction is created in the current directory and has the same name as the fileset being linked. After the command completes, the new junction appears as an ordinary directory, except that the user is not allowed to unlink or delete it with the **rmdir** command it. The user can use the **mv** command on the directory to move to a new location in the parent fileset, but the **mv** command is not allowed to move the junction to a different fileset.

For complete usage information, see the topic *mmlinkfileset command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Unlinking a fileset

A junction to a fileset is removed with the **mmunlinkfileset** command, which unlinks the fileset only from the active directory namespace. The linked or unlinked state of a fileset in a snapshot is unaffected. The unlink fails if there are files open in the fileset, unless the **-f** option is specified. The root fileset cannot be unlinked.

After issuing the **mmunlinkfileset** command, the fileset can be re-linked to a different parent using the **mmlinkfileset** command. Until the fileset is re-linked, it is not accessible.

Note: If run against a file system that has an unlinked fileset, **mmapplypolicy** will not traverse the unlinked fileset.

Attention: If you are using the IBM Spectrum Protect Backup-Archive client you must use caution when you unlink filesets that contain data backed up by IBM Spectrum Protect. IBM Spectrum Protect tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to IBM Spectrum Protect that you deleted the contents of the fileset. Therefore, the IBM Spectrum Protect Backup-Archive client inactivates the data on the IBM Spectrum Protect server which may result in the loss of backup data during the expiration process.

For complete usage information, see the topic *mmunlinkfileset command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Changing fileset attributes

To change a fileset's junction, you have to first unlink the fileset using the **mmunlinkfileset** command, and then create the new junction using the **mmlinkfileset** command.

To change the attributes of an existing fileset, including the fileset name, use the **mmchfileset** command.

Note: In an HSM-managed file system, moving or renaming migrated files between filesets will result in recalling of the date from the IBM Spectrum Protect server.

For complete usage information, see the topics *mmchfileset command*, *mmlinkfileset command*, and *mmunlinkfileset command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Displaying fileset information

Fileset status and attributes are displayed with the **mmlsfileset** command.

Some of the attributes displayed include:
- Name of the fileset.
- Fileset identifier of the fileset.

- Junction path to the fileset.
- Status of the fileset.
- Root inode number of the fileset.
- Path to the fileset (if linked).
- Inode space.
- User provided comments (if any).

For complete usage information, see the topic *mmlsfileset command* in the *IBM Spectrum Scale: Command and Programming Reference*.

To display the name of the fileset that includes a given file, run the **mmlsattr** command and specify the **-L** option. For complete usage information, see the topic *mmlsattr command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Immutability and appendOnly features

To prevent files from being changed or deleted unexpectedly, GPFS provides immutability and appendOnly restrictions.

Applying immutability and appendOnly restrictions to individual files or to directories

You can apply immutability and appendOnly restrictions either to individual files within a fileset or to a directory.

An immutable file cannot be changed or renamed. An appendOnly file allows append operations, but not delete, modify, or rename operations.

An immutable directory cannot be deleted or renamed, and files cannot be added or deleted under such a directory. An appendOnly directory allows new files or subdirectories to be created with 0 byte length; all such new created files and subdirectories are marked as appendOnly automatically.

The **immutable** flag and the **appendOnly** flag can be set independently. If both immutability and appendOnly are set on a file, immutability restrictions will be in effect.

To set or unset these attributes, use the following command options:

mmchattr -i {yes | no}
> Sets or unsets a file to or from an immutable state.

>> **-i yes**   Sets the **immutable** attribute of the file to **yes**.

>> **-i no**   Sets the **immutable** attribute of the file to **no**.

mmchattr -a {yes | no}
> Sets or unsets a file to or from an appendOnly state.

>> **-a yes**   Sets the **appendOnly** attribute of the file to **yes**.

>> **-a no**   Sets the **appendOnly** attribute of the file to **no**.

Note: Before an immutable or appendOnly file can be deleted, you must change it to mutable or set appendOnly to **no** (by using the **mmchattr** command).

Storage pool assignment of an immutable or appendOnly file can be changed; an immutable or appendOnly file is allowed to transfer from one storage pool to another.

To display whether or not a file is immutable or appendOnly, issue this command:

```
mmlsattr -L myfile
```

The system displays information similar to the following:

```
file name:            myfile
metadata replication: 2 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    sp1
fileset name:         root
snapshot name:
creation Time:        Wed Feb 22 15:16:29 2012
Misc attributes:      ARCHIVE
```

The effects of file operations on immutable and appendOnly files

Once a file has been set as immutable or appendOnly, the following file operations and attributes work differently from the way they work on regular files:

delete An immutable or appendOnly file cannot be deleted.

modify/append
>An appendOnly file cannot be modified, but it can be appended. An immutable file cannot be modified or appended.
>
>**Note:** The immutable and appendOnly flag check takes effect after the file is closed; therefore, the file can be modified if it is opened before the file is changed to immutable.

mode An immutable or appendOnly file's mode cannot be changed.

ownership, acl
>These attributes cannot be changed for an immutable or appendOnly file.

extended attributes
>These attributes cannot be added, deleted, or modified for an immutable or appendOnly file.

timestamp
>The timestamp of an immutable or appendOnly file can be changed.

directory
>If a directory is marked as immutable, no files can be created, renamed, or deleted under that directory. However, a subdirectory under an immutable directory remains mutable unless it is explicitly changed by **mmchattr**.
>
>If a directory is marked as appendOnly, no files can be renamed or deleted under that directory. However, 0 byte length files can be created.

The following table shows the effects of file operations on an immutable file or an appendOnly file:

Table 40. The effects of file operations on an immutable file or an appendOnly file

Operation	immutable	appendOnly
Add, **delete**, **modify**, or **rename**	No	No
Append	No	Yes
Change ownership, **mode**, or **acl**	No	No
Change atime, **mtime**, or **ctime**	Yes	Yes

Table 40. The effects of file operations on an immutable file or an appendOnly file (continued)

Operation	immutable	appendOnly
Add, **delete**, or **modify extended attributes**	Disallowed by external methods such as **setfattr**. Allowed internally for **dmapi**, **directio**, and others.	Same as for immutable.
Create a file under an immutable or appendOnly directory	No	Yes, 0 byte length only
Rename or **delete** a file under an immutable or appendOnly directory	No	No
Modify a mutable file under an immutable directory	Yes	Not applicable
Set an immutable file back to mutable	Yes	Not applicable
Set an appendOnly file back to a non-appendOnly state	Not applicable	Yes

Fileset-level integrated archive manager (IAM) modes

You can modify the file-operation restrictions that apply to the immutable files in a fileset by setting an integrated archive manager (IAM) mode for the fileset. The following table shows the effects of each of the IAM modes.

Note: To set an IAM mode for a fileset, issue the `mmchfileset` command with the --iam-mode parameter. For more information, see the topic *mmchfileset* in the *IBM Spectrum Scale: Command and Programming Reference*.

Table 41. IAM modes and their effects on file operations on immutable files

File operation	Regular mode	Advisory mode	Noncompliant mode	Compliant mode	
Modify	No	No	No	No	
Append	No	No	No	No	
Rename	No	No	No	No	
Change ownership, acl	No	No	No	No	
Change mode	No	No	No	No	

File operation	Regular mode	Advisory mode	Noncompliant mode	Compliant mode	
Change **atime**, **mtime**, **ctime**	Yes	**mtime** and **ctime** can be changed. **atime** is overloaded by expiration time. Expiration time can be changed by using the **mmchattr --expiration-time** command(alternatively **mmchattr -E**) or touch. You can see the expiration time by using stat as **atime**.	Same as advisory mode	Same as advisory mode	
Add, delete, or modify extended attributes.	Not allowed for external methods such as **setfattr**. Allowed internally for dmapi, directio, and etc.	Yes	Yes	Yes	
Create, rename, or delete under an immutable directory	No	No	No	No	
Modify mutable files under an immutable directory.	Yes	Yes	Yes	Yes	
Retention rule enforced	No retention rule, cannot delete immutable files	No	Yes	Yes	
Set ExpirationTime backwards	Yes	Yes	Yes	No	
Delete an immutable file	No	Yes, always	Yes, only when expired	Yes, only when expired	
Set an immutable file back to mutable	Yes	No	No	No	
Allow hardlink	No for immutable or appendOnly files. Yes for other files.	No	No	No	

Table 41. IAM modes and their effects on file operations on immutable files (continued)

File operation	Regular mode	Advisory mode	Noncompliant mode	Compliant mode	
Rename or delete a non-empty directory	Yes for rename. No for delete.	No for rename. Yes for delete.	No for rename. Yes for delete only if the immutable file has expired.	No for rename. Yes for delete only if the immutable file has expired.	
Remove user write permission to change a file to immutable	No	Yes	Yes	Yes	
Display expiration time instead of **atime** for stat call	No	Yes	Yes	Yes	
Set a directory to be immutable	Yes	No	No	No	

Chapter 26. Creating and maintaining snapshots of file systems

A snapshot of an entire GPFS file system can be created to preserve the contents of the file system at a single point in time. Snapshots of the entire file system are also known as global snapshots. The storage overhead for maintaining a snapshot is keeping a copy of data blocks that would otherwise be changed or deleted after the time of the snapshot.

Snapshots of a file system are read-only; changes can only be made to the active (that is, normal, non-snapshot) files and directories.

The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time that the snapshot was created. Snapshots also provide an online backup capability that allows easy recovery from common problems such as accidental deletion of a file, and comparison with older versions of a file.

Notes:

1. Because snapshots are not copies of the entire file system, they should not be used as protection against media failures. For information about protection against media failures, see the topic *Recoverability considerations* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

2. Fileset snapshots provide a method to create a snapshot of an independent fileset instead of the entire file system. For more information about fileset snapshots, see "Fileset-level snapshots" on page 404.

3. A snapshot of a file creates a new file that captures the user data and user attributes from the original. The snapshot file is independent from the original file. For DMAPI managed file systems, the snapshot of a file is not automatically managed by DMAPI, regardless of the state of the original file. The DMAPI attributes from the original file are not inherited by the snapshot. For more information about DMAPI restrictions for GPFS, see the *IBM Spectrum Scale: Command and Programming Reference*.

4. When snapshots are present, deleting files from the active file system does not always result in any space actually being freed up; rather, blocks may be pushed to the previous snapshot. In this situation, the way to free up space is to delete the oldest snapshot. Before creating new snapshots, it is good practice to ensure that the file system is not close to being full.

5. The use of clones functionally provides writable snapshots. See Chapter 27, "Creating and managing file clones," on page 421.

Management of snapshots of a GPFS file system includes:

- "Creating a snapshot"
- "Listing snapshots" on page 414
- "Restoring a file system from a snapshot" on page 415
- "Reading a snapshot with the policy engine" on page 416
- "Linking to a snapshot" on page 416
- "Deleting a snapshot" on page 417

Creating a snapshot

Use the **mmcrsnapshot** command to create a snapshot of an entire GPFS file system at a single point in time. Snapshots appear in the file system tree as hidden subdirectories of the root.

Global snapshots appear in a subdirectory in the root directory of the file system, whose default name is **.snapshots**. If you prefer to access snapshots from each directory rather than traversing through the root

directory, you can use an invisible directory to make the connection by issuing the **mmsnapdir** command. See "Linking to a snapshot" on page 416 for more information.

A snapshot of the file system, *Device*, is identified by a *SnapshotName* name on the **mmcrsnapshot** command. For example, given the file system **fs1** to create a snapshot **snap1**, enter:

```
mmcrsnapshot fs1 snap1
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap1 created with id 1.
```

Before issuing the command, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If a second snapshot were to be created at a later time, the first snapshot would remain as is. A snapshot can be made only of an active file system, not of an existing snapshot. The following command creates another snapshot of the same file system:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap2 created with id 2.
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3

/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userA/file2
/fs1/.snapshots/snap2/userA/file3
```

For complete usage information, see the topic *mmcrsnapshot command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Listing snapshots

Use the **mmlssnapshot** command to display existing snapshots of a file system and their attributes.

The **-d** option displays the amount of storage used by a snapshot. GPFS quota management does not take the data blocks used to store snapshots into account when reporting on and determining if quota limits have been exceeded. This is a slow operation and its usage is suggested for problem determination only.

For example, to display the snapshot information for the file system **fs1** with additional storage information, issue this command:

```
mmlssnapshot fs1 -d
```

The system displays information similar to:

```
Snapshots in file system fs1: [data and metadata in KB]
Directory               SnapId   Status       Created                         Data  Metadata
snap1                   1        Valid        Fri Oct 17 10:56:22 2003          0     512
```

For complete usage information, see the topic *mmlssnapshot command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Restoring a file system from a snapshot

Use the **mmrestorefs** command to restore user data and attribute files in an active file system from a snapshot.

Prior to issuing the **mmrestorefs** command, ensure that the file system is mounted. When restoring from an independent fileset snapshot, ensure that the fileset is in linked state.

Existing snapshots, including the one being used in the restore, are not modified by the **mmrestorefs** command. To obtain a snapshot of the restored file system, you must issue the **mmcrsnapshot** command to capture it before issuing the **mmrestorefs** command again.

As an example, suppose that you have a directory structure similar to the following:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If the directory **userA** is then deleted, the structure becomes similar to this:

```
/fs1/file1
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

The directory **userB** is then created using the inode originally assigned to **userA**, and another snapshot is taken:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap2 created with id 2.
```

The resulting directory structure is similar to the following:

```
/fs1/file1
/fs1/userB/file2b
/fs1/userB/file3b
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
```

```
/fs1/.snapshots/snap1/userA/file3
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

The file system is then restored from **snap1**:

```
mmrestorefs fs1 snap1
```

The resulting directory structure is similar to the following:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

For complete usage information, see the topic *mmrestorefs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Reading a snapshot with the policy engine

You can use the policy engine to read the contents of a snapshot for backup purposes. The **mmapplypolicy** command provides the **-S** option to specify the snapshot during a policy run. Instead of matching rules to the active file system, the policy engine matches the rules against files in the snapshot.

Notes:

1. Snapshots are read-only. Policy rules such as MIGRATE or DELETE that make changes or delete files cannot be used with a snapshot.

2. An instance of **mmapplypolicy** can only scan one snapshot. Directing it at the .snapshots directory itself will result in a failure.

For complete usage information, see the topic *mmapplypolicy command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Linking to a snapshot

Snapshot root directories appear in a special **.snapshots** directory under the file system root.

If you prefer to link directly to the snapshot rather than always traverse the root directory, you can use the **mmsnapdir** command with the **-a** option to add a **.snapshots** subdirectory to all directories in the file system. These **.snapshots** subdirectories will contain a link into the corresponding directory for each snapshot that includes the directory in the active file system.

Unlike **.snapshots** in the root directory, however, the **.snapshots** directories added by the **-a** option of the **mmsnapdir** command are invisible in the sense that the **ls** command or **readdir()** function does not return **.snapshots**. This is to prevent recursive file system utilities such as **find** or **tar** from entering into the snapshot tree for each directory they process. For example, if you enter **ls -a /fs1/userA**, the **.snapshots** directory is not listed. However, you can enter **ls /fs1/userA/.snapshots** or **cd /fs1/userA/.snapshots** to confirm that **.snapshots** is present. If a user wants to make one of their snapshot directories more visible, it is suggested to create a symbolic link to **.snapshots**.

The inode numbers that are used for and within these special **.snapshots** directories are constructed dynamically and do not follow the standard rules. These inode numbers are visible to applications

through standard commands, such as **stat**, **readdir**, or **ls**. The inode numbers reported for these directories can also be reported differently on different operating systems. Applications should not expect consistent numbering for such inodes.

Specifying the **-r** option on the **mmsnapdir** command reverses the effect of the **-a** option, and reverts to the default behavior of a single **.snapshots** directory in the root directory.

The **-s** option allows you to change the name of the **.snapshots** directory. For complete usage information, see the topic *mmsnapdir command* in the *IBM Spectrum Scale: Command and Programming Reference*.

To illustrate this point, assume that a GPFS file system called **fs1**, which is mounted at **/fs1**, has one snapshot called **snap1**. The file system might appear similar to this:

```
/fs1/userA/file2b
/fs1/userA/file3b
/fs1/.snapshots/snap1/userA/file2b
/fs1/.snapshots/snap1/userA/file3b
```

To create links to the snapshots from each directory, and instead of **.snapshots**, use the name **.links**, enter:

```
mmsnapdir fs1 -a -s .links
```

After the command completes, the directory structure would appear similar to:

```
/fs1/userA/file2b
/fs1/userA/file3b
/fs1/userA/.links/snap1/file2b
/fs1/userA/.links/snap1/file3b

/fs1/.links/snap1/userA/file2b
/fs1/.links/snap1/userA/file3b
```

To delete the links, issue:

```
mmsnapdir fs1 -r
```

After the command completes, the directory structure is similar to the following:

```
/fs1/userA/file2b
/fs1/userA/file3b

/fs1/.links/snap1/userA/file2b
/fs1/.links/snap1/userA/file3b
```

For complete usage information, see the topic *mmsnapdir command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Deleting a snapshot

Use the **mmdelsnapshot** command to delete GPFS snapshots of a file system.

For example, to delete **snap1** for the file system **fs1**, enter:

```
mmdelsnapshot fs1 snap1
```

The output is similar to this:

```
Invalidating snapshot files...
Deleting snapshot files...
 100.00 % complete on Tue Feb 28 10:40:59 2012
Delete snapshot snap1 complete, err = 0
```

For complete usage information, see the topic *mmdelsnapshot command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Managing snapshots using IBM Spectrum Scale GUI

Use **Files** > **Snapshots** page in the IBM Spectrum Scale GUI to manage snapshots through GUI.

Snapshots can be used in environments where multiple recovery points are necessary. A snapshot can be taken of file system or fileset data and then the data can be recovered from the snapshot if the production data becomes unavailable.

Note:
* Snapshots are read-only; changes can be made only to the normal and active files and directories, not to the snapshot.
* When a snapshot of an independent fileset is taken, only nested dependent filesets are included in the snapshot.

Scheduling snapshot creation by using snapshot rules

You can either manually create the snapshots or create snapshot rules to automate the snapshot creation and retention.

To manually create a snapshot, click **Create Snapshot** in the Snapshots page and enter the required details under the **Manual** tab of the Create Snapshot window. Click **Create** after entering the details.

By creating a snapshot rule, you can automate the snapshot creation and retention. That is, in a snapshot rule you can specify a frequency in which the snapshots must be created and the number of snapshots that must be retained for a period. The retention policy helps to avoid unwanted storage of snapshots that result in waste of storage resources.

Retention policy has the following parameters:
* Frequency of snapshot creation
* Number of most recent snapshots to be retained. The most recent snapshot is identified based on the frequency of snapshot creation.
* Number of days for which you need to keep the latest snapshot of each day.
* Number of weeks for which you need to keep the latest snapshot of each week.
* Number of months for which you need to keep the latest snapshot of each month.

Example scenario for retention policy

The following table provides an example for the values that are specified against these parameters.

Table 42. Example for retention period

Frequency	Minute	Number of most recent snapshots	Keep latest snapshots for			
			Hours	Days	Weeks	Months
Hourly	1	2	2	6	2	3

Based on the this retention rule, the following snapshots are created and retained on March 20, 2016 at 06:10 AM:

Table 43. Example - Time stamp of snapshots that are retained based on the retention policy

Time stamp	Condition based on which snapshot is retained
December 31 (Thursday, 23:01 AM)	Keep latest snapshot for last 3 months
January 31 (Sunday, 23:01 AM)	Keep latest snapshot for last 3 months

Table 43. Example - Time stamp of snapshots that are retained based on the retention policy (continued)

Time stamp	Condition based on which snapshot is retained
February 29 (Monday, 23:01 AM)	Keep latest snapshot for last 3 months
March 5 (Saturday, 23:01 AM)	Keep latest snapshot for last 2 weeks
March 12 (Saturday, 23:01 AM)	Keep latest snapshot for last 2 weeks
March 14 (Monday, 23:01 AM)	Keep latest snapshot for last 6 days
March 15 (Tuesday, 23:01 AM)	Keep latest snapshot for last 6 days
March 16 (Wednesday, 23:01 AM)	Keep latest snapshot for last 6 days
March 17 (Thursday, 23:01 AM)	Keep latest snapshot for last 6 days
March 18 (Friday, 23:01 AM)	Keep latest snapshot for last 6 days
March 19 (Saturday, 23:01 AM)	Keep latest snapshot for last 6 days
March 20 (Sunday, 5:01 AM)	Keep 2 most recent
March 20 (Sunday, 6:01 AM)	Keep 2 most recent

According to this rule, 13 snapshots are retained on March 20, 2016 at 06:10 AM.

To schedule snapshot creation and retention, perform the following steps:
1. Go to **Files > Snapshots**.
2. Click **Create Snapshot**.
3. In the Create Snapshot window, enter the path of the file system or independent fileset for which you need to create snapshots.
4. In the **Snapshot name** field, specify the name of the snapshot.
5. Click **Snapshot Rules**.
6. Click **Create Rule** to schedule the snapshot creation and retention. Create Snapshot Rule window is displayed.
7. In the **Name** field, type the name of the snapshot scheduling rule.
8. In the **Frequency** field, select the frequency in which you need to create snapshot. You need to enter some more details based on the value that is selected in the **Frequency** field. For example, if value selected is **Multiple Times an Hour**, select the minutes of the hour in which you need to create snapshots.
9. In the **Retention** fields, specify the number of snapshots that must be retained in a time period.
10. In the **Prefix** field, specify a prefix to be added with the name of the snapshots that are created with this rule.
11. Click **OK** to save the changes.

If you do not specify a name for the snapshot, the default name is given. The default snapshot ID is generated at the creation time by using the format "*@GMT-yyyy.MM.dd-HH.mm.ss*". If this option is given and the "*@GMT-date-time*" format is omitted, then this snapshot will not be identifiable by Windows VSS and the file restore is not possible by that method. Avoid white spaces, double and single quotation marks, the parentheses (), the star *, forward slash /, and backward slash \.

Deleting snapshots

To manually delete the snapshots, right-click the snapshot from the Snapshots page and select **Delete**. The snapshots that are automatically created based on the snapshot creation rule, are deleted automatically based on the retention period specified in the rule. When the condition for deletion is met, the GUI immediately startS to delete the snapshot candidates.

Creating and deleting peer and RPO snapshots

The peer and recovery point objective (RPO) snapshots are used in the AFM and AFM DR configurations to ensure data integrity and availability. When a peer snapshot is taken, it creates a snapshot of the cache fileset and then queues a snapshot creation at the home site. This ensures application consistency at both cache and home sites. The RPO snapshot is a type of peer snapshot that is used in the AFM DR setup. It is used to maintain consistency between the primary and secondary sites in an AFM DR configuration.

Use the **Create Peer Snapshot** option in the **Files** > **Snapshots** page to create peer snapshots. You can view and delete these peer snapshots from the Snapshots page and also from the detailed view of the **Files** > **Active File Management** page.

Chapter 27. Creating and managing file clones

A file clone is a writable snapshot of an individual file. File clones can be used to provision virtual machines by creating a virtual disk for each machine by cloning a common base image. A related usage is to clone the virtual disk image of an individual machine as part of taking a snapshot of the machine state.

Cloning a file is similar to creating a copy of a file, but the creation process is faster and more space efficient because no additional disk space is consumed until the clone or the original file is modified. Multiple clones of the same file can be created with no additional space overhead. You can also create clones of clones.

Management of file clones in a GPFS file system includes:
- "Creating file clones"
- "Listing file clones" on page 422
- "Deleting file clones" on page 423
- "File clones and disk space management" on page 423
- "File clones and snapshots" on page 423

Creating file clones

File clones can be created from a regular file or a file in a snapshot using the **mmclone** command.

Creating a file clone from a regular file is a two-step process using the **mmclone** command with the **snap** and **copy** keywords:

1. Issue the **mmclone snap** command to create a read-only snapshot of the file to be cloned. This read-only snapshot becomes known as the clone parent. For example, the following command creates a clone parent called **snap1** from the original file **file1**:

   ```
   mmclone snap file1 snap1
   ```

   Alternately, if only one file is specified with the **mmclone snap** command, it will convert the file to a read-only clone parent without creating a separate clone parent file. When using this method to create a clone parent, the specified file cannot be open for writing or have hard links. For example, the following command converts **file1** into a clone parent.

   ```
   mmclone snap file1
   ```

2. Issue the **mmclone copy** command to create a writable clone from a clone parent. For example, the following command creates a writable file clone called **file2** from the clone parent **snap1**:

   ```
   mmclone copy snap1 file2
   ```

Creating a file clone where the source is in a snapshot only requires one step using the **mmclone** command with the **copy** keyword. For example, the following command creates a writable file clone called **file3.clone** from a file called **file3** in a snapshot called **snap2**:

```
mmclone copy /fs1/.snapshots/snap2/file3 file3.clone
```

In this case, **file3** becomes the clone parent.

Note: Extended attributes of clone parents are not passed along to file clones.

After a clone has been created, the clone and the file that it was cloned from are interchangeable, which is similar to a regular copy (**cp**) command. The file clone will have a new inode number and attributes that can be modified independently of the original file.

Additional clones can be created from the same clone parent by issuing additional **mmclone copy** commands, for example:

```
mmclone copy snap1 file3
```

File clones of clones can also be created, as shown in the following example:

```
mmclone snap file1 snap1
mmclone copy snap1 file2
echo hello >> file2
mmclone snap file2 snap2
mmclone copy snap2 file3
```

The echo command updates the last block of file clone **file2**. When **file2** is snapped to **snap2**, the **mmclone snap** operation is performed as described previously. When a block in **file3** is read, the clone parent inode is found first. For the case of the last block, with the **hello** text, the disk address will be found in **snap2**. However, for other blocks, the disk address will be found in **snap1**.

For complete usage information, see the topic *mmclone command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Listing file clones

Use the **mmclone** command to display status for specified files.

The **show** keyword of the **mmclone** command provides a report to determine the current status of one or more files. When a file is a clone, the report will show the parent inode number. When a file was cloned from a file in a snapshot, **mmclone show** displays the snapshot and fileset information.

Consider the following scenario:

1. The **ls** command is issued to show all **img** files in the current directory:

   ```
   ls -ils *.img
   ```

   The system displays output similar to the following:

   ```
   148485 5752576 -rw-r--r-- 1 root root 21474836480 Jan  9 16:19 test01.img
   ```

2. A file clone is then created with the following commands:

   ```
   mmclone snap test01.img base.img
   mmclone copy base.img test02.img
   ```

3. After the file clone is created, the **mmclone show** command is issued to show information about all **img** files in the current directory:

   ```
   mmclone show *.img
   ```

   The system displays output similar to the following:

   ```
   Parent  Depth  Parent inode   File name
   ------  -----  --------------  ---------
      yes    0                    base.img
       no    1           148488  test01.img
       no    1           148488  test02.img
   ```

4. A subsequent **ls** command would display output similar to the following:

   ```
   # ls -ils *.img
   148488 5752576 -rw-r--r-- 3 root root 21474836480 Jan  9 16:25 base.img
   148485       0 -rw-r--r-- 1 root root 21474836480 Jan  9 16:19 test01.img
   148480       0 -rw-r--r-- 1 root root 21474836480 Jan  9 16:25 test02.img
   ```

For complete usage information, see the topic *mmclone command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Deleting file clones

There is no explicit GPFS command available for deleting file clones. File clones can be deleted using a regular delete (**rm**) command. Clone parent files cannot be deleted until all file clone copies of the parent have been deleted and all open file handles to them have been closed.

Note: There is a brief period of time, immediately following the deletion of the file clone copies, when deletion of the parent can fail because the clone copy deletions are still running in the background.

Splitting file clones from clone parents

Use the **mmclone** command to split a file clone from a clone parent.

File clones can be split from their clone parents in one of two ways:

- Using the **mmclone redirect** command to split the file clone from the immediate clone parent only. The clone child remains a file clone, but the clone parent can be deleted.
- Using the **mmclone split** command to split the file clone from all clone parents. This converts the former clone child to a regular file. The clone parent does not change.

For complete usage information, see the topic *mmclone command* in the *IBM Spectrum Scale: Command and Programming Reference*.

File clones and disk space management

File clones have several considerations that are related to disk space management.

- Replication and storage pools

  Each file clone has its own inode, so attributes and permissions for each clone can be set independently. For example, timestamps (atime, mtime, and ctime) are maintained separately for each clone. Clone parent attributes must be changed separately. If different clones have different values for replication or storage pool, it is not possible for every one of the clones to have all data blocks readable through that clone to be replicated and placed consistent with its replication and pool settings. Thus, changes to replication and storage pool only apply to blocks added to the clone and leave the clone parent unchanged.

- Clone ownership, block counts, and quotas

  Creating a clone parent requires read access to the file being cloned. The person creating the clone parent does not have to be the owner of the original file, but will become the owner of the new clone parent. The block count and disk quota of the original file will be transferred to the new clone parent inode and then set to zero in the original file. Any blocks allocated by copy-on-write of a clone file will be added to the block count in the clone inode, with quota charged against the owner of the file. If even a single byte of data in a clone child is changed, the entire block will be copied from the parent.

- Clones and DMAPI

  Clone parent files and clone copy files will be preserved across migrations and recalls to and from tape.

File clones and snapshots

When a snapshot is created and a file clone is subsequently updated, the previous state of the file clone will be saved in the snapshot.

When reading a file clone in the snapshot, the system will distinguish between the states of the clone:

- The data block has not been modified since the snapshot was taken, so look for the data in the same file in the next most recent snapshot or active file system.
- The file clone was updated, which indicates that the corresponding data block should be found in the clone parent. The system will look for the data in the clone parent within the same snapshot.

When a snapshot has file clones, those file clones should be deleted or split from their clone parents prior to deleting the snapshot. See "Deleting file clones" on page 423 and "Splitting file clones from clone parents" on page 423 for more information. A policy file can be created to help determine if a snapshot has file clones. See "File clones and policy files" for more information.

File clones and policy files

Policy files can be created to examine clone attributes.

The following clone attributes can be examined in a policy file:
- The depth of the clone tree.
- If file is an immutable clone parent.
- The fileset ID of the clone parent.
- The inode number of the clone parent for the file.
- If the clone parent is in a snapshot.
- The snapshot ID of the clone parent.

See "File attributes in SQL expressions" on page 371 for more information about the clone attributes available for policy files.

The following example shows a policy file that can be created for displaying clone attributes for all files:

```
RULE EXTERNAL LIST 'x' EXEC ''
RULE 'nonClone' LIST 'x' SHOW('nonclone') WHERE Clone_Parent_Inode IS NULL
RULE 'normalClone' LIST 'x' SHOW(
  'inum ' || varchar(Clone_Parent_Inode) ||
  ' par ' || varchar(Clone_Is_Parent) ||
  ' psn ' || varchar(Clone_Parent_Is_Snap) ||
  ' dep ' || varchar(Clone_Depth))
  WHERE Clone_Parent_Inode IS NOT NULL AND Clone_Parent_Is_Snap == 0
RULE 'snapClone' LIST 'x' SHOW(
  'inum ' || varchar(Clone_Parent_Inode) ||
  ' par ' || varchar(Clone_Is_Parent) ||
  ' psn ' || varchar(Clone_Parent_Is_Snap) ||
  ' dep ' || varchar(Clone_Depth) ||
  ' Fid ' || varchar(Clone_Parent_Fileset_Id) ||
  ' snap ' || varchar(Clone_Parent_Snap_Id))
  WHERE Clone_Parent_Inode IS NOT NULL AND Clone_Parent_Is_Snap != 0
```

If this policy file was called **pol.file**, the following command would display the clone attributes:

```
mmapplypolicy fs0 -P pol.file -I defer -f pol -L 0
```

Chapter 28. Scale Out Backup and Restore (SOBAR)

Scale Out Backup and Restore (SOBAR) is a specialized mechanism for data protection against disaster only for GPFS file systems that are managed by IBM Spectrum Protect for Space Management.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

To protect a file system against disaster the following steps must be taken to ensure all data is safely stored in a second location:

1. Record the file system configuration with the `mmbackupconfig` command.
2. Ensure all file data is *pre-migrated* (see "Pre-migrating files with external storage pools" on page 398 for more information).
3. Perform a metadata image backup with the `mmimgbackup` command.

The `mmbackupconfig` command must be run prior to running the `mmimgbackup` command. No changes to file system configuration, filesets, quotas, or other settings should be done between running the `mmbackupconfig` command and the `mmimgbackup` command. To recover from a disaster, the `mmrestoreconfig` command must be run prior to running the `mmimgrestore` command. The file system being restored must have the same inode size and metadata block size as the file system that was backed up. Use the `mmrestoreconfig -F QueryResultFile` option to create the QueryResultFile. Use the example of the `mmcrfs` command within the QueryResultFile to recreate your file system. After restoring the image data and adjusting quota settings, the file system can be mounted read-write, and the HSM system re-enabled to permit file data recall. Users may be permitted to access the file system, and/or the system administrator can manually recall file data with the IBM Spectrum Protect for Space Management command `dsmrecall`.

These commands cannot be run from a Windows node.

Backup procedure with SOBAR

This section provides a detailed example of the backup procedure used with SOBAR.

Throughout these procedures, the sample file system used is called **smallfs**. Where appropriate, replace this value with your file system name.

1. Backup the cluster configuration information.

   The cluster configuration must be backed up by the administrator. The minimum cluster configuration information needed is: IP addresses, node names, roles, quorum and server roles, cluster-wide configuration settings from **mmchconfig**, cluster manager node roles, remote shell configuration, mutual **ssh** and **rsh** authentication setup, and the cluster UID. More complete configuration information can be found in the **mmsdrfs file** and **CCR**.

2. Preserve disk configuration information.

   Disk configuration must also be preserved in order to recover a file system. The basic disk configuration information needed, for a backup intended for disaster recovery, is the number of disk volumes that were previously available and the sizes of those volumes. In order to recover from a complete file system loss, at least as much disk space as was previously available will be needed for restoration. It is only feasible to restore the image of a file system onto replacement disks if the disk volumes available are of similar enough sizes to the originals that all data can be restored to the new disks. At a minimum, the following disk configuration information is needed:

   - Disk device names
   - Disk device sizes

- The number of disk volumes
- NSD server configuration
- Disk RAID configurations
- Failure group designations
- The **mmsdrfs** file contents

3. Backup the GPFS file system configuration information.

   In addition to the disks, the file system built on those volumes has configuration information that can be captured using the **mmbackupconfig** command. This information includes block size, replication factors, number and size of disks, storage pool layout, filesets and junction points, policy rules, quota information, and a number of other file system attributes. The file system configuration information can be backed up into a single file using a command similar to the following:

   ```
   mmbackupconfig smallfs -o /tmp/smallfs.bkpcfg.out925
   ```

   Be sure to copy the temporary file that is created by the preceding command to a secure location so that it can be retrieved and used during a disaster recovery.

4. Pre-migrate all newer file data into secondary storage.

   File contents in a space-managed GPFS will reside in secondary storage managed by the HSM. In the case of IBM Spectrum Protect HSM, disk and tape pools will typically hold the offline images of migrated files. HSM can also be used to pre-migrate all newer file data into secondary storage, so that all files will have either a migrated or pre-migrated status (**XATTR**) recorded, and their current contents are copied or updated into the secondary storage. The IBM Spectrum Protect command **dsmmigrate** can be used as follows:

   ```
   dsmmigrate -Premigrate -Recursive /smallfs
   ```

   To optionally check the status of the files that were pre-migrated with the previous command, use the following command:

   ```
   dsmls /smallfs/*
   ```

5. Create a global snapshot of the live file system, to provide a quiescent image for image backup, using a command similar to the following:

   ```
   mmcrsnapshot smallfs smallfssnap
   ```

6. Choose a staging area in which to save the GPFS metadata image files.

   The image backup process stores each piece of the partial file system image backup in its own file in the shared work directory typically used by policy runs. These files can become quite large depending on the number of files in the file system. Also, because the file system holding this shared directory must be accessible to every node participating in the parallel backup task, it might also be a GPFS file system. It is imperative that the staging directory chosen be accessible to both the **tsapolicy** archiver process and the IBM Spectrum Protect Backup-Archive client. This staging directory is specified with the **-g** option of the **mmimgbackup** command.

7. Backup the file system image.

   The following command will back up an image of the GPFS metadata from the file system using a parallel policy run with the default IBM Spectrum Protect backup client to backup the file system metadata image:

   ```
   mmimgbackup smallfs -S smallfssnap -g /u/user/backup -N aixnodes
   ```

   The metadata of the file system, the directories, inodes, attributes, symlinks, and so on are all captured in parallel by using the archive module extension feature of the **mmapplypolicy** command. After completing the parallel execution of the policy-driven archiving process, a collection of image files in this format will remain. These image files are gathered by the **mmimgbackup** command and archived to IBM Spectrum Protect automatically.

   If you are using the **-N** *nodes* option, it is a good idea to use the same operating system when running **mmimgbackup**. Also, the directory that was created with the **-g** *GlobalWorkDirectory* option to store the image files must exist and must be accessible from all the nodes that are specified.

8. After the image backup is complete, delete the snapshot used for backup with the following command:

```
mmdelsnapshot smallfs smallfssnap
```

Restore procedure with SOBAR

This section provides a detailed example of the restore procedure used with SOBAR.

In order to restore a file system, the configuration data stored from a previous run of **mmbackupconfig** and the image files produced from **mmimgbackup** must be accessible.

Throughout these procedures, the sample file system used is called **smallfs**. Where appropriate, replace this value with your file system name.

1. Restore the metadata image files from **mmimgbackup** and the backup configuration data from **mmbackupconfig** with a **dsmc** command similar to the following:

```
dsmc restore -subdir=yes /u/user/backup/8516/
```

2. Retrieve the base file system configuration information.

   Use the **mmrestoreconfig** command to generate a configuration file, which contains the details of the former file system:

```
mmrestoreconfig Device -i InputFile -F QueryResultFile
```

3. Recreate NSDs if they are missing.

   Using the output file generated in the previous step as a guide, the administrator might need to recreate NSD devices for use with the restored file system. In the output file, the **NSD configuration** section contains the NSD information; for example:

```
######## NSD configuration #################
## Disk descriptor format for the mmcrnsd command.
## Please edit the disk and desired name fields to match
## your current hardware settings.
##
## The user then can uncomment the descriptor lines and
## use this file as input to the -F option.
#
# %nsd:
#   device=DiskName
#   nsd=nsd8
#   usage=dataAndMetadata
#   failureGroup=-1
#   pool=system
#
```

   If changes are needed, edit the file in a text editor and follow the included instructions to use it as input to the **mmcrnsd** command, then issue the following command:

```
mmcrnsd -F StanzaFile
```

4. Recreate the base file system.

   The administrator must recreate the initial file system. The output query file specified in the previous commands can be used as a guide. The following example shows the section of this file that is needed when recreating the file system:

```
######### File system configuration #############
## The user can use the predefined options/option values
## when recreating the file system.  The option values
## represent values from the backed up file system.
#
# mmcrfs FS_NAME NSD_DISKS -j cluster -k posix -Q yes -L 4194304 --disable-fastea
  -T /fs2 -A no --inode-limit 278016#
#
```

```
# When preparing the file system for image restore, quota
# enforcement must be disabled at file system creation time.
# If this is not done, the image restore process will fail.
```

Do one of the following to recreate the file system:

- Edit the output file. Uncomment the **mmcrfs** command, and specify the appropriate file system name and NSD disk(s). Remove the **-Q** option to ensure quotas are not enabled. Save the changes and run the file as a shell script:

  `sh OutputFile`

- From the command line, issue an **mmcrfs** command similar to the one in the output file, but specify the appropriate file system name and NSD disk(s). Do not specify the **-Q** option to ensure quotas are not enabled. The inode size and metadata block size must be the same in the file system in order to be restored as is in the original file system.

5. Restore essential file system configuration.

   Using the **mmrestoreconfig** command, the essential file system configuration can be restored to the file system that was just created in the previous step. Quota is disabled in this step because the quota system must remain inactive until after the file system image has been restored. Filesets will also be restored and linked, if necessary, using a method specific for image restore. The **--image-restore** option should be used to restore the configuration data in the proper format for SOBAR; for example:

   `mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 --image-restore`

6. Mount the file system in read-only mode for image restore with the following command:

   `mmmount smallfs -o ro`

7. Perform the image restore; for example:

   `mmimgrestore smallfs /u/user/backup/8516/mmPolicy.8551.D4D85229`

8. To optionally display the restored file system structure, use the following command:

   `ls -l /smallfs/*`

   The system displays information similar to the following:

   ```
   -rw-r--r-- 1 root root 1024 Sep 25 11:34 /smallfs/1Kfile.1
   -rw-r--r-- 1 root root 1024 Sep 25 11:34 /smallfs/1Kfile.2
   -rwxr--r-- 1 root root  238 Sep 25 11:34 /smallfs/generateChksums*
   ```

9. Unmount the file system with the following command:

   `mmumount smallfs`

10. Restore quota configuration.

    If any quota enforcement was used in the prior file system, it can be restored now using the **mmrestoreconfig** command. This step will not enable quotas if they were not in use at the time of the configuration backup. To restore the quota configuration, issue a command similar to the following:

    `mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 -Q only`

11. Mount the file system in read-write mode with the following command:

    `mmmount smallfs`

12. Delete the unusable HSM directory.

    The **.SpaceMan** directory contains file stubs from the former space management control information. This directory must be deleted prior to restarting HSM management. Use the following command:

    `rm -rf /smallfs/.SpaceMan`

13. To optionally restart HSM, use the following command:

    `dsmmigfs restart`

14. Resume HSM management on the newly reconstructed file system, to resume managing disk space and to permit recall of files, with the following command:

    `dsmmigfs add /smallfs`

15. To optionally display the managed file system from HSM, use the following command:
```
dsmls /smallfs/*
```

All files are currently in the migrate state.

16. To optionally begin recalling files by forcing a specific recall, use the following command:
```
dsmrecall -Recursive /smallfs/*
```

Chapter 29. Data Mirroring and Replication

The ability to detect and quickly recover from a massive hardware failure is of paramount importance to businesses that make use of real-time data processing systems.

GPFS provides a number of features that facilitate the implementation of highly-available GPFS environments capable of withstanding catastrophic hardware failures. By maintaining a replica of the file system's data at a geographically-separate location, the system sustains its processing using the secondary replica of the file system in the event of a total failure in the primary environment.

On a high level, a disaster-resilient GPFS cluster is made up of two or three, distinct, geographically-separate hardware sites operating in a coordinated fashion. Two of the sites consist of GPFS nodes and storage resources holding a complete replica of the file system. If a third site is active, it consists of a single node and a single disk used as a tiebreaker for GPFS quorum. In the event of a catastrophic hardware failure that disables the operation of an entire site, and assuming the tiebreaker site remains operational, file system services fail over to the remaining subset of the cluster and continue serving the data using the replica of the file system that survived the disaster. However, if the tiebreaker fails during the disaster, the remaining number of nodes and disks is insufficient to satisfy the quorum rules and the surviving site loses access to the GPFS file system. A manual procedure is needed to instruct GPFS to disregard the existing quorum assignments and continue operating with whatever resources are available.

The secondary replica is maintained by one of several methods:

- Synchronous mirroring utilizing GPFS replication.

    The data and metadata replication features of GPFS are used to implement synchronous mirroring between a pair of geographically-separate sites. The use of logical replication-based mirroring offers a generic solution that relies on no specific support from the disk subsystem beyond the basic ability to read and write data blocks..

- Synchronous mirroring utilizing storage-based replication.

    Hardware replication creates persistent mirroring relationship between pairs of Logical Units (LUNs) on two subsystems connected over SAN or LAN links. All updates performed on the set of primary, source, or LUNs appear in the same order on the secondary, target, or disks in the target subsystem. Hardware replication provides for an exact bitwise replica of the content of the source as seen at the time of the failure on the target if the source volume fails. A range of technologies can be used to provide synchronous replication such as Metro Mirror on DS8000® or Storwize® or Synchronous Remote Mirroring on XIV®.

- Asynchronous mirroring utilizing GPFS-based replication.

    Asynchronous replication functionality provides a similar crash consistent copy of data as synchronous replication but in normal operation the secondary copy of data will lag behind the primary by some period of time. For more information, see the topic *AFM-based Asynchronous Disaster Recovery (AFM DR)* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- Asynchronous mirroring utilizing storage-based replication.

    Asynchronous replication functionality provides a similar crash consistent copy of data as synchronous replication but in normal operation the secondary copy of data will lag behind the primary by some time. A range of technologies can be used to provide asynchronous replication such as Global Mirror on DS8000 or Storwize or Asynchronous Remote Mirroring on XIV.

- Point in time copy using storage-based functionality.

    Periodic point-in-time copies of the file system are taken using the functionality such as FlashCopy® on the DS8000 or Storwize or Snapshot on XIV. This copy could be used as a source of a complete file

system consistent backup to be taken to a remote site or could be used in conjunction with other replication capabilities to use for isolated testing of disaster recovery procedures.

The primary advantage of both synchronous mirroring methods is the minimization of the risk of permanent data loss. Both methods provide two consistent, up-to-date replicas of the file system, each available for recovery if the other one fails. However, inherent to all solutions that synchronously mirror data over a wide area network link is the latency penalty that is induced by the replicated write I/Os. This makes both synchronous mirroring methods prohibitively inefficient for certain types of performance-oriented applications of where there is a longer distance between sites. The asynchronous method effectively eliminates this penalty but in a situation where the primary site is lost, there might be updates that have not yet been transferred to the secondary site. Asynchronous replication will still provide a crash consistent and restartable copy of the primary data.

General considerations for using storage replication with GPFS

This topic describes the general considerations that need to be followed for using storage replication with IBM Spectrum Scale.

Different storage-level replication capabilities are available on both IBM and non-IBM storage systems. IBM provides storage-level replication functionality on the following platforms:

- The DS8000 provides synchronous replication with Metro Mirror and asynchronous replication with Global Mirror. Three and four site replication topologies are also possible by combining these functions. For more information, see **IBM DS8000 series V7.2 documentation** at http://www-01.ibm.com/ support/knowledgecenter/HW213_7.2.0/com.ibm.storage.ssic.help.doc/f2c_ichomepage.htm.
- The Storwize family of storage systems also provides a synchronous replication capability with Metro Mirror and has two versions of asynchronous replication called Global Mirror and Global Mirror with Change Volumes. Point in Time copy functionality is provided by FlashCopy. For more information, see **IBM Storwize V7000** at http://www-01.ibm.com/support/knowledgecenter/ST3FR7/welcome.
- The XIV provides both synchronous and asynchronous Remote Replication and also provides point in time copy functionality referred to as Snapshot. For more information, see **IBM XIV Storage System documentation** at http://www-01.ibm.com/support/knowledgecenter/STJTAG/ com.ibm.help.xivgen3.doc/xiv_kcwelcomepage.html.

Note: In this document, synchronous replication is referred to as Metro Mirror, asynchronous replication is referred to as Global Mirror, and point in time copy functionality is referred to as FlashCopy.

Data integrity and the use of consistency groups

Disk based replication technologies provide a crash consistent copy of the replicated data. This means that the data that is not committed to the second site when a failure occurs may not be saved. However, a volume can be recovered to a recent point in time when all of the data was consistent.

A group of volumes that share a common recovery point is commonly called a consistency group. The storage controller ensures that after a failure, all of the volumes within a consistency group are recovered to the same point in time.

When using a storage-based replication with IBM Spectrum Scale, it is important to ensure that all the NSD's in a file system are contained within the same consistency group. This way the metadata NSD's are always in sync with the data NSD's after a failure.

Handling multiple versions of IBM Spectrum Scale data

This topic provides description on handling multiple versions of data in IBM Spectrum Scale.

The primary copy of a GPFS file system and a hardware replicated copy cannot coexist in the same GPFS cluster. A node can mount either the original copy of the file system or one of its replicas, but not both. This restriction has to do with the current implementation of the NSD-to-LUN mapping mechanism, which scans all locally attached disks, searching for a specific value (the NSD ID) at a particular location on disk. If both the original volume and a hardware replica are visible to a particular node, these disks would appear to GPFS as distinct devices with identical NSD IDs.

For this reason, users are asked to zone their SAN configurations such that at most one replica of any given GPFS disk is visible from any node. That is, the nodes in your production cluster should have access to the disks that make up the actual file system but should not see the disks that hold the replicated copies, whereas the backup server should see the replication targets but not the originals.

Alternatively, you can use the **nsddevices** user exit located in `/var/mmfs/etc/` to explicitly define the subset of the locally visible disks to be accessed during the NSD device scan on the local node.

The following procedure is used to define an **nsddevices** user exit file to instruct GPFS to use a specific disk **diskA1** rather than other copies of this device, which might also be available:

```
echo "echo diskA1 hdisk" > /var/mmfs/etc/nsddevices chmod 744 /var/mmfs/etc/nsddevices
```

Refer to the prolog of `/usr/lpp/mmfs/samples/nsddevices.samples` for detailed instructions on the usage of **nsddevices**.

Continuous Replication of IBM Spectrum Scale data

This topic provides a brief description on replication of IBM Spectrum Scale data.

Synchronous mirroring with GPFS replication

In a configuration utilizing GPFS replication, a single GPFS cluster is defined over three geographically-separate sites consisting of two production sites and a third tiebreaker site. One or more file systems are created, mounted, and accessed concurrently from the two active production sites.

The data and metadata replication features of GPFS are used to maintain a secondary copy of each file system block, relying on the concept of disk failure groups to control the physical placement of the individual copies:

1. Separate the set of available disk volumes into two failure groups. Define one failure group at each of the active production sites.
2. Create a replicated file system. Specify a replication factor of 2 for both data and metadata.

When allocating new file system blocks, GPFS always assigns replicas of the same block to distinct failure groups. This provides a sufficient level of redundancy allowing each site to continue operating independently should the other site fail.

GPFS enforces a node quorum rule to prevent multiple nodes from assuming the role of the file system manager in the event of a network partition. Thus, a majority of quorum nodes must remain active in order for the cluster to sustain normal file system usage. Furthermore, GPFS uses a quorum replication algorithm to maintain the content of the file system descriptor (one of the central elements of the GPFS metadata). When formatting the file system, GPFS assigns some number of disks (usually three) as the descriptor replica holders that are responsible for maintaining an up-to-date copy of the descriptor. Similar to the node quorum requirement, a majority of the replica holder disks must remain available at all times to sustain normal file system operations. This file system descriptor quorum is internally controlled by the GPFS daemon. However, when a disk has failed due to a disaster you must manually inform GPFS that the disk is no longer available and it should be excluded from use.

Considering these quorum constraints, it is suggested that a third site in the configuration fulfill the role of a tiebreaker for the node and the file system descriptor quorum decisions. The tiebreaker site consists of:

1. A single quorum node

   As the function of this node is to serve as a tiebreaker in GPFS quorum decisions, it does not require normal file system access and SAN connectivity. To ignore disk access errors on the tiebreaker node, enable the **unmountOnDiskFail** configuration parameter through the **mmchconfig** command. When enabled, this parameter forces the tiebreaker node to treat the lack of disk connectivity as a local error, resulting in a failure to mount the file system, rather that reporting this condition to the file system manager as a disk failure.

2. A single network shared disk

   The function of this disk is to provide an additional replica of the file system descriptor file needed to sustain quorum should a disaster cripple one of the other descriptor replica disks. Create a network shared disk over the tiebreaker node's internal disk defining:

   - the local node as an NSD server
   - the disk usage as **descOnly**

     The **descOnly** option instructs GPFS to only store file system descriptor information on the disk.

This three-site configuration is resilient to a complete failure of any single hardware site. Should all disk volumes in one of the failure groups become unavailable, GPFS performs a transparent failover to the remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention. While nothing prevents you from placing the tiebreaker resources at one of the active sites, to minimize the risk of double-site failures it is suggested you install the tiebreakers at a third, geographically distinct location.

Important: Note the following good practices:
- In an environment that is running synchronous mirroring using GPFS replication:
  - Do not designate a tiebreaker node as a manager node.
  - Do not mount the file system on the tiebreaker node. To avoid unexpected mounts, you can create the following file with any content on the tiebreaker node:

    `/var/mmfs/etc/ignoreAnyMount.<file_system_name>`

    In the following example, the file system is `fs1`:

    `/var/mmfs/etc/ignoreAnyMount.fs1`

    See the article "File System Management" in IBM developerWorks® at https://www.ibm.com/ developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System %20(GPFS)/page/File%20System%20Management.

    Note: If you create an `ignoreAnyMount.<file_system_name>` file, you cannot manually mount the file system on the tiebreaker node.

  If you do not follow these practices, an unexpected file system unmount can occur during site failures, because of the configuration of the tiebreaker node and the **unmountOnDiskFail** option.
- In a stretch cluster environment, designate at least one quorum node from each site as a manager node. During site outages, the quorum nodes can take over as manager nodes.

Note: There are no special networking requirements for this configuration. For example:
- You do not need to create different subnets.
- You do not need to have GPFS nodes in the same network across the two production sites.
- The production sites can be on different virtual LANs (VLANs).

Limitation: If the Object protocol is deployed on the cluster and the CES networks of two production sites cannot communicate with each other, you must change the Object Ring configuration to use the CES IP addresses of only one of the production sites. Follow the procedure that is described in the topic "Configuration of object for isolated node and network groups" on page 280.

The high-level organization of a replicated GPFS cluster for synchronous mirroring where all disks are directly attached to all nodes in the cluster is shown in Figure 13. An alternative to this design would be to have the data served through designated NSD servers.

With GPFS release 4.1.0, a new, more fault-tolerant configuration mechanism has been introduced as the successor for the server-based mechanisms. The server-based configuration mechanisms consist of two configuration servers specified as the primary and secondary cluster configuration server. The new configuration mechanism uses all specified quorum nodes in the cluster to hold the GPFS configuration and is called CCR (Clustered Configuration Repository). The CCR is used by default during cluster creation unless the CCR is explicitly disabled. The mmlscluster command reports the configuration mechanism in use in the cluster.

The following sections describe the differences regarding disaster recovery for the two configuration mechanisms.
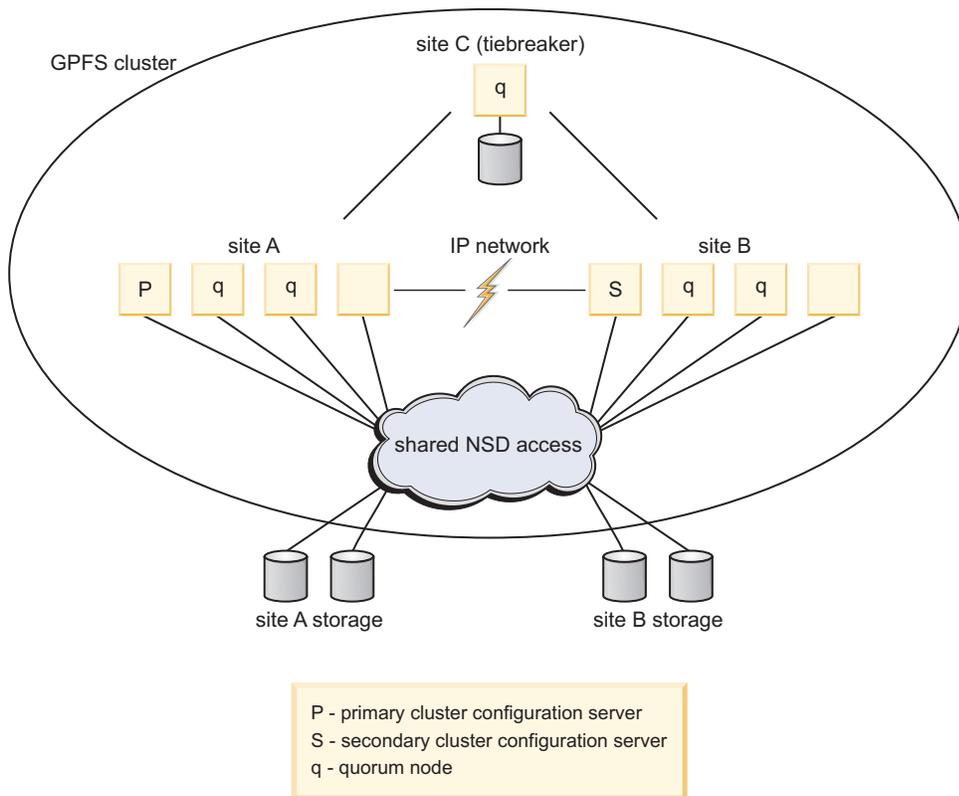
Figure 13. Synchronous mirroring utilizing GPFS replication

Setting up IBM Spectrum Scale synchronous replication

See an example of how to set up synchronous IBM Spectrum Scale replication to recover from a site failure.

Synchronous replication is described in the topic and in Figure 1 of that topic.

This example is based on the following configuration:

Site A

- Nodes: **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**
- Disk devices: **diskA1** and **diskA2**. These devices are SAN-attached and accessible from all the nodes at site **A** and site **B**.

Site B

- Nodes: **nodeB001**, **nodeB002**, **nodeB003**, **node B004**
- Disk devices: **diskB1** and **diskB2**. These devices are SAN-attached and accessible from all the nodes at site **A** and site **B**.

Site C Note that site **C** contains only one node, which will be defined as a quorum node and a client node.

- Nodes: **nodeC**
- Disks: **diskC**. This disk is an NSD defined on an internal disk of **nodeC** and is directly accessible only from site **C**.

1. Create an IBM Spectrum Scale cluster with the `mmcrcluster` command and a node file.

    a. Create a node file that is named `clusterNodes` with the following contents:

    ```
    nodeA001:quorum-manager
    nodeA002:quorum-manager
    nodeA003:quorum-manager
    nodeA004:client
    nodeB001:quorum-manager
    nodeB002:quorum-manager
    nodeB003:quorum-manager
    nodeB004:client
    nodeC:quorum-client
    ```

    b. Issue the following command to create the cluster:

    ```
    mmcrcluster -N clusterNodes
    ```

    Note: The cluster is created with the Cluster Configuration Repository (CCR) enabled. This option is the default on IBM Spectrum Scale v4.1 or later.

2. Issue the following command to enable the `unmountOnDiskFile` attribute on **nodeC**:

    ```
    mmchconfig unmountOnDiskFail=yes -N nodeC
    ```

    Enabling this attribute prevents false disk errors in the SAN configuration from being reported to the file system manager.

    Important: In a synchronous replication environment, the following rules are good practices:

    - The following rules apply to **nodeC**, which is the only node on site **C** and is also a client node and a quorum node:
        - Do not designate the **nodeC** as a manager node.
        - Do not mount the file system on **nodeC**.

          To avoid unexpected mounts, create the following empty file on **nodeC**:

          ```
          /var/mmfs/etc/ignoreAnyMount.<file_system_name>
          ```

          For example, if the file system is `fs0`, create the following empty file:

          ```
          /var/mmfs/etc/ignoreAnyMount.fs0
          ```

          See the article "File System Management" in IBM developerWorks at https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20(GPFS)/page/File%20System%20Management.

          Note: If you create an `ignoreAnyMount.<file_system_name>` file, you cannot manually mount the file system on **nodeC**.

If you do not follow these practices, an unexpected file system unmount can occur during site failures, because of the configuration of **nodeC** and the **unmountOnDiskFail** option.

- In the sites that do not contain a single quorum client node (sites **A** and **B** in this example), designate at least one quorum node from each site as a manager node. During a site outage, a quorum node can take over as a manager node.

3. Create a set of network shared disks (NSDs) for the cluster.

   a. Create the stanza file `clusterDisks` with the following NSD stanzas:

   ```
   %nsd: device=/dev/diskA1
   servers=nodeA002,nodeA003
   usage=dataAndMetadata
   failureGroup=1
   %nsd: device=/dev/diskA2
   servers=nodeA003,nodeA002
   usage=dataAndMetadata
   failureGroup=1
   %nsd: device=/dev/diskB1
   servers=nodeB002,nodeB003
   usage=dataAndMetadata
   failureGroup=2
   %nsd: device=/dev/diskB2
   servers=nodeB003,nodeB002
   usage=dataAndMetadata
   failureGroup=2
   %nsd: device=/dev/diskC1
   servers=nodeC
   usage=descOnly
   failureGroup=3
   ```

   Important: Note that the stanzas make the following failure group assignments:
   - The disks at site **A** are assigned to failure group 1.
   - The disks at site **B** are assigned to failure group 2.
   - The disk that is local to **nodeC** is assigned to failure group 3.

   b. Issue the following command to create the NSDs:

   ```
   mmcrnsd –F clusterDisks
   ```

   c. Issue the following command to verify that the network shared disks are created:

   ```
   mmlsnsd -m
   ```

   The command should display output like the following:

   ```
   Disk name     NSD volume ID      Device       Node name       Remarks
   ---------------------------------------------------------------------------
    gpfs1nsd     0972445B416BE502   /dev/diskA1   nodeA002        server node
    gpfs1nsd     0972445B416BE502   /dev/diskA1   nodeA003        server node
    gpfs2nsd     0972445B416BE509   /dev/diskA2   nodeA002        server node
    gpfs2nsd     0972445B416BE509   /dev/diskA2   nodeA003        server node
    gpfs3nsd     0972445F416BE4F8   /dev/diskB1   nodeB002        server node
    gpfs3nsd     0972445F416BE4F8   /dev/diskB1   nodeB003        server node
    gpfs4nsd     0972445F416BE4FE   /dev/diskB2   nodeB002        server node
    gpfs4nsd     0972445F416BE4FE   /dev/diskB2   nodeB003        server node
    gpfs5nsd     0972445D416BE504   /dev/diskC1   nodeC           server node
   ```

4. Issue the following command to start IBM Spectrum Scale on all the nodes of the cluster:

   ```
   mmstartup -a
   ```

5. Create a file system `fs0` with default replication for metadata (`-m 2`) and data (`-r 2`). Issue the following command:

   ```
   mmcrfs /gpfs/fs0 fs0 –F clusterDisks -m 2 -r 2
   ```

6. Mount the file system `fs0` on all the cluster nodes at site **A** and site **B**.

7. This step is optional and for ease of use. Issue the following three commands to create node classes for sites **A**, **B**, and **C**:

```
mmcrnodeclass gpfs.siteA -N prt001st001,prt002st001,prt003st001,prt004st001,nsd001st001,nsd002st001
mmcrnodeclass gpfs.siteB -N prt008st001,prt007st001,prt006st001,prt005st001,nsd004st001,nsd003st001
mmcrnodeclass gpfs.siteC -N nsd005st001
```

You can now use node class names with IBM Spectrum Scale commands ("mm" commands) to recover sites easily after a cluster failover and failback. For example, with the following command you can bring down all the nodes on site **B** with one parameter, rather than having to pass all the node names for site **B** into the command:

```
mmshutdown -N gpfs.siteB
```

For information on the recovery procedure, see "Failback with temporary loss using the Clustered Configuration Repository (CCR) configuration mechanism" on page 441.

The cluster is configured with synchronous replication to recover from a site failure.

Steps to take after a disaster when using Spectrum Scale replication

Utilizing GPFS replication allows for *failover* to the surviving site without disruption of service as long as both the remaining site and the tiebreaker site remain functional. It remains in this state until a decision is made to restore the operation of the affected site by executing the *failback* procedure. If the tiebreaker site is also affected by the disaster and is no longer operational, GPFS quorum is broken and manual intervention is required to resume file system access.

Existing quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements:
1. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to non-quorum nodes. Issue the **mmchnode --nonquorum** command.
2. To relax file system descriptor quorum, temporarily eliminate the failed disks from the group of disks from which the GPFS daemon uses to write the file system descriptor file to. Issue the **mmfsctl exclude** command for each of the failed disks.

While the GPFS cluster is in a failover state, it is suggested that no changes to the GPFS configuration be made. If the server-based configuration mechanism is in use, changes to your GPFS configuration require both cluster configuration servers to be operational. If both servers are not operational, the sites would have distinct, and possibly inconsistent, copies of the GPFS **mmsdrfs** configuration data file. While the servers can be migrated to the surviving site, it is best to avoid this step if the disaster does not leave the affected site permanently disabled.

If it becomes absolutely necessary to modify the GPFS configuration while in failover mode, for example to relax quorum, you must ensure that all nodes at the affected site are powered down and left in a stable inactive state. They must remain in such state until the decision is made to execute the failback procedure. As a means of precaution, we suggest disabling the GPFS autoload option on all nodes to prevent GPFS from bringing itself up automatically on the affected nodes should they come up spontaneously at some point after a disaster.

Failover to the surviving site:

Following a disaster, which failover process is implemented depends upon whether or not the tiebreaker site is affected.

Failover without the loss of tiebreaker site C

The proposed three-site configuration is resilient to a complete failure of any single hardware site. Should all disk volumes in one of the failure groups become unavailable, GPFS performs a transparent failover to the remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention.

Failover with the loss of tiebreaker site C with server-based configuration in use

If both site **A** and site **C** fail:

1. Shut the GPFS daemon down on the surviving nodes at site **B**, where the file **gpfs.siteB** lists all of the nodes at site **B**:

   `mmshutdown -N gpfs.siteB`

2. If it is necessary to make changes to the configuration, migrate the primary cluster configuration server to a node at site **B**:

   `mmchcluster -p nodeB002`

3. Relax node quorum by temporarily changing the designation of each of the failed quorum nodes to non-quorum nodes:

   `mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC`

4. Relax file system descriptor quorum by informing the GPFS daemon to migrate the file system descriptor off of the failed disks:

   `mmfsctl fs0 exclude -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"`

5. Restart the GPFS daemon on the surviving nodes:

   `mmstartup -N gpfs.siteB`

6. Mount the file system on the surviving nodes at site B.

Failover with the loss of tiebreaker site C with Clustered Configuration Repository (CCR) in use

If both site A and site C fail:

1. Shut the GPFS daemon down on the surviving nodes at site B , where the file gpfs.siteB lists all of the nodes at site B :

   `mmdsh -N gpfs.siteB /usr/lpp/mmfs/bin/mmshutdown`

2. Changing (downgrading) the quorum assignments when half or more of the quorum nodes are no longer available at site B using the –- force option :

   `mmchnode –-nonquorum -N nodeA001,nodeA002,nodeA003,nodeC --force`

3. Relax file system descriptor quorum by informing the GPFS daemon to migrate the file system descriptor off of the failed disks:

   `mmfsctl fs0 exclude -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"`

4. Restart the GPFS daemon on the surviving nodes:

   `mmstartup -N gpfs.siteB`

5. Mount the file system on the surviving nodes at site B.

Make no further changes to the quorum designations at site B until the failed sites are back on line and the following failback procedure has been completed.

Do not shut down the current set of nodes on the surviving site B and restart operations on the failed sites A and C. This will result in a non-working cluster.

Failback procedures:

Which failback procedure you follow depends upon whether the nodes and disks at the affected site have been repaired or replaced.

If the disks have been repaired, you must also consider the state of the data on the failed disks:

• For nodes and disks that have been repaired and *you are certain* the data on the failed disks has not been changed, follow either:

  – *failback with temporary loss and no configuration changes*
  – *failback with temporary loss and configuration changes*

- If the nodes have been replaced and either the disks have been replaced or repaired, and *you are not certain* the data on the fail disks has not been changed, follow the procedure for *failback with permanent loss*.

Delayed failures: In certain failure cases the loss of data may not be immediately apparent. For example, consider this sequence of events:

1. Site **B** loses connectivity with sites **A** and **C**.
2. Site **B** then goes down due to loss of node quorum.
3. Sites **A** and **C** remain operational long enough to modify some of the data on disk but suffer a disastrous failure shortly afterwards.
4. Node and file system descriptor quorums are overridden to enable access at site **B**.

Now the two replicas of the file system are inconsistent and the only way to reconcile these copies during recovery is to:

1. Remove the damaged disks at sites **A** and **C**.
2. Either replace the disk and format a new NSD or simply reformat the existing disk if possible.
3. Add the disk back to the file system, performing a full resynchronization of the file system's data and metadata and restore the replica balance using the **mmrestripefs** command.

Failback with temporary loss and no configuration changes:

If the outage was of a temporary nature and your configuration has not been altered, it is a simple process to fail back to the original state.

After all affected nodes and disks have been repaired and *you are certain* the data on the failed disks has not been changed:

1. Start GPFS on the repaired nodes where the file **gpfs.sitesAC** lists all of the nodes at sites **A** and **C**:
   ```
   mmstartup -N gpfs.sitesAC
   ```
2. Restart the affected disks. If more than one disk in the file system is down, they must all be started at the same time:
   ```
   mmchdisk fs0 start -a
   ```

Failback with temporary loss and configuration changes in the server-based configuration:

If the outage was of a temporary nature and your configuration has been altered, follow this procedure to fail back to the original state in case primary and secondary configuration servers are in use.

After all affected nodes and disks have been repaired and *you are certain* that the data on the failed disks has not been changed:

1. Ensure that all nodes have the latest copy of the `mmsdrfs` file:
   ```
   mmchcluster -p LATEST
   ```

   For more information about the `mmsdrfs` file, see *Recovery from loss of GPFS cluster configuration data file* in the *IBM Spectrum Scale: Problem Determination Guide*.
2. Migrate the primary cluster configuration server back to site **A**:
   ```
   mmchcluster -p nodeA001
   ```
3. Restore node quorum designations at sites **A** and **C**:
   ```
   mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```
4. Start GPFS on the repaired nodes where the file **gpfs.sitesAC** lists all of the nodes at sites **A** and **C**:
   ```
   mmstartup -N gpfs.sitesAC
   ```
5. Restore the file system descriptor quorum by informing the GPFS to include the repaired disks:
   ```
   mmfsctl fs0 include -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```

6. Bring the disks online and restripe the file system across all disks in the cluster to restore the initial replication properties:

```
mmchdisk fs0 start -a
mmrestripefs fs0 -b
```

The **-r** flag may be used on the **mmrestripefs** command instead.

Failback with temporary loss using the Clustered Configuration Repository (CCR) configuration mechanism:

If the outage was of a temporary nature and your configuration has been altered, follow this procedure to failback to the original state, in case the Clustered Configuration Repository (CCR) configuration scheme is in use.

After all affected nodes and disks have been repaired and you are certain the data on the failed disks has not been changed, complete the following steps.

1. Shut down the GPFS daemon on the surviving nodes at site B , and on the former failed and now recovered sites A and C , where the file `gpfs.siteB` lists all of the nodes at site B and the file `gpfs.siteA` lists all of the nodes at site A and the tiebreaker node at site C:

```
mmshutdown -N gpfs.siteB
mmshutdown -N gpfs.siteA
mmshutdown -N nodeC
```

2. Restore original node quorum designation for the tiebreaker site C at site B and start GPFS on site C:

```
mmstartup -N gpfs.siteB
mmchnode --quorum -N nodeC
mmstartup -N nodeC
```

3. Restore original node quorum designation for site A at site B and start GPFS on site A:

```
mmchnode --quorum -N nodeA001,nodeA002,nodeA003
mmstartup -N gpfs.siteA
```

4. Restore the file system descriptor quorum by informing the GPFS to include the repaired disks:

```
mmumount fs0 -a;mmfsctl fs0 include -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
```

5. Mount the file system on all nodes at sites A and B.

   Note: Do not allow the failed sites A and C to come online at the same time or when site B is unavailable or not functional.

6. Bring the disks online and restripe the file system across all disks in the cluster to restore the initial replication properties:

```
mmchdisk fs0 start -a
mmrestripefs fs0 -b
```

The **-r** flag can be used on the `mmrestripefs` command instead.

Failback with permanent loss:

If an outage is of a permanent nature, follow steps to remove and replace the failed resources, and then resume the operation of GPFS across the cluster.

1. Remove the failed resources from the GPFS configuration
2. Replace the failed resources, then add the new resources into the configuration
3. Resume the operation of GPFS across the entire cluster

Assume that sites **A** and **C** have had permanent losses. To remove all references of the failed nodes and disks from the GPFS configuration and replace them:

Procedure when the server-based configuration scheme is in use

1. To remove the failed resources from the GPFS configuration:

a. If as part of the failover process, *you did not* migrate the primary cluster configuration server, migrate the server to node **nodeB002** at site B:

```
mmchcluster –p nodeB002
```

b. Delete the failed disks from the GPFS configuration:

```
mmdeldisk fs0 "gpfs1nsd;gpfs2nsd;gpfs5nsd"
mmdelnsd "gpfs1nsd;gpfs2nsd;gpfs5nsd"
```

c. Delete the failed nodes from the GPFS configuration:

```
mmdelnode -N nodeA001,nodeA002,nodeA003,nodeA004,nodeC
```

2. If there are new resources to add to the configuration:

a. Add the new nodes at sites **A** and **C** to the cluster where the file **gpfs.sitesAC** lists the new nodes:

```
mmaddnode -N gpfs.sitesAC
```

b. Ensure that all nodes have the latest copy of the **mmsdrfs** file:

```
mmchcluster -p LATEST
```

c. Migrate the primary cluster configuration server back to site **A**:

```
mmchcluster -p nodeA001
```

d. Start GPFS on the new nodes

```
mmstartup -N gpfs.sitesAC
```

e. Prepare the new disks for use in the cluster, create the NSDs using the original disk descriptors for site **A** contained in the file **clusterDisksAC**:

```
%nsd: device=/dev/diskA1
    servers=nodeA002,nodeA003
    usage=dataAndMetadata
    failureGroup=1

%nsd: device=/dev/diskA2
    servers=nodeA003,nodeA002
    usage=dataAndMetadata
    failureGroup=1

%nsd: device=/dev/diskC1
    servers=nodeC
    usage=descOnly

failureGroup=3mmcrnsd -F clusterDisksAC
```

f. Add the new NSDs to the file system specifying the **-r** option to rebalance the data on all disks:

```
mmadddisk fs0 -F clusterDisksAC -r
```

Procedure when Clustered Configuration Repository (CCR) is in use

1. To remove the failed resources from the GPFS configuration:

a. Delete the failed disks from the GPFS configuration:

```
mmdeldisk fs0 "gpfs1nsd;gpfs2nsd;gpfs5nsd"
```
```
mmdelnsd "gpfs1nsd;gpfs2nsd;gpfs5nsd"
```

b. Delete the failed nodes from the GPFS configuration:

```
mmdelnode -N nodeA001,nodeA002,nodeA003,nodeA004,nodeC
```

2. If there are new resources to add to the configuration:

a. Add the new nodes at sites **A** and **C** to the cluster where the file **gpfs.sitesAC** lists the new nodes:

```
mmaddnode -N gpfs.sitesAC
```

b. Restore original quorum node assignments at site B:

```
mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
```

c. Start GPFS on the new nodes

```
mmstartup -N gpfs.sitesAC
```

d. Prepare the new disks for use in the cluster, create the NSDs using the original disk descriptors for site **A** contained in the file **clusterDisksAC**:

```
%nsd: device=/dev/diskA1
 servers=nodeA002,nodeA003
 usage=dataAndMetadata
 failureGroup=1

%nsd: device=/dev/diskA2
 servers=nodeA003,nodeA002
 usage=dataAndMetadata
 failureGroup=1

%nsd: device=/dev/diskC1
 servers=nodeC
 usage=descOnly

 failureGroup=3mmcrnsd -F clusterDisksAC
```

e. Add the new NSDs to the file system specifying the **-r** option to rebalance the data on all disks:

```
mmadddisk fs0 -F clusterDisksAC -r
```

Synchronous mirroring utilizing storage based replication

This topic describes synchronous mirroring utilizing storage-based replication.

Synchronous replication in the storage layer continuously updates a secondary (target) copy of a disk volume to match changes made to a primary (source) volume. A pair of volumes are configured in a replication relationship, during which all write operations performed on the source are synchronously mirrored to the target device.

The synchronous replication protocol guarantees that the secondary copy is constantly up-to-date by ensuring that the primary copy is written only if the primary storage subsystem received an acknowledgment that the secondary copy has been written. The paired volumes typically reside on two distinct and geographically separated storage systems communicating over a SAN or LAN link.

Most synchronous replication solutions provide a capability to perform an incremental resynchronization of data when switching between primary and secondary storage systems. After the failure of the primary volume (or the failure of the entire storage subsystem or site), users perform a failover, which suspends the relationship between the given pair of volumes and turns the target volume into a primary. When a volume enters the suspended state, a modification bitmap is established to keep track of the write operations performed on that volume to allow for an efficient resynchronization.

Once the operation of the original primary volume has been restored, a failback is executed to resynchronize the content of the two volumes. The original source volume is switched to the target mode, after which all modified data tracks (those recorded in the modification bitmap) are copied from the original target disk. The volume pair can then be suspended again and a similar process performed to reverse the volumes' roles, thus bringing the pair into its initial state.

A GPFS cluster using hardware-based replication can be established in two manners:
- A single GPFS cluster encompassing two sites and an optional tiebreaker site
- Two distinct GPFS clusters

An active-active Spectrum Scale cluster

In an active-active cluster, a single GPFS cluster contains two active sites and an optional tiebreaker site.

The high-level organization of an active/active GPFS cluster using hardware replication is illustrated in Figure 14 on page 444. A single GPFS cluster is created over three sites. The data is mirrored between two active sites with a cluster configuration server residing at each site and a tiebreaker quorum node installed at the third location. The presence of an optional tiebreaker node allows the surviving site to

satisfy the node quorum requirement with no additional intervention. Without the tiebreaker, the failover procedure requires an additional administrative command to relax node quorum and allow the remaining site to function independently. Furthermore, the nodes at the recovery site have direct disk paths to the primary site's storage.

The GPFS configuration resides either on the two configuration server (primary and secondary), when the cluster has been created with the Clustered Configuration Repository (CCR) disable option (**mmcrcluster**), or on each quorum node, when the cluster has Clustered Configuration Repository (CCR) enabled, or on the primary/secondary, when the Clustered Configuration Repository (CCR) is disabled.
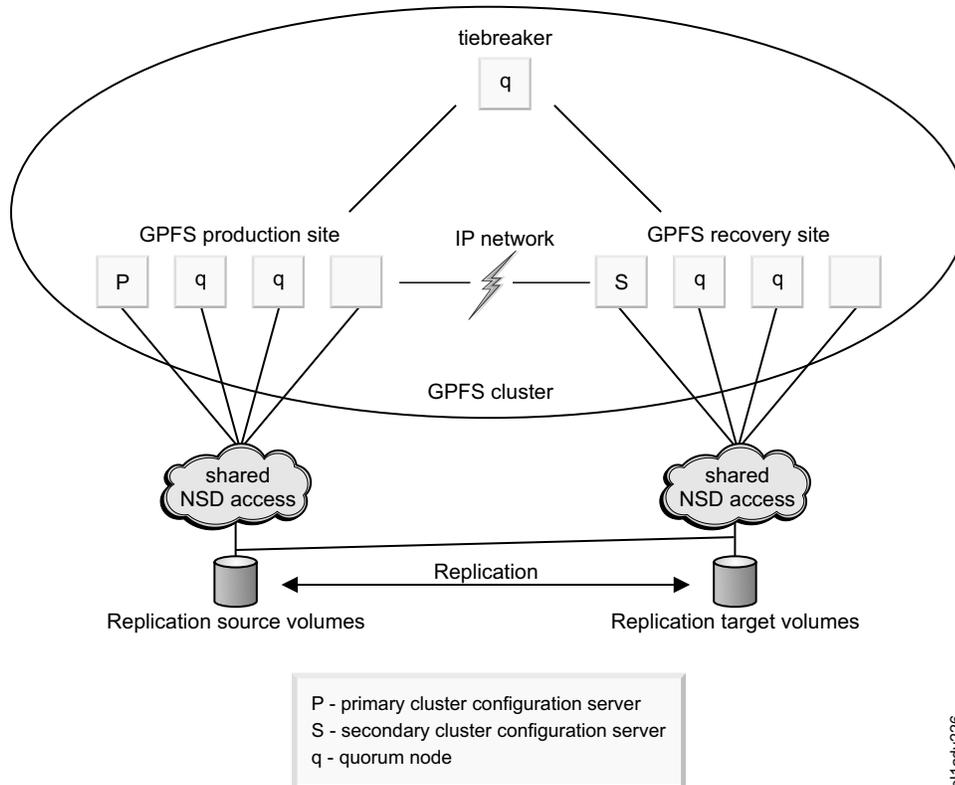
Figure 14. A synchronous active-active replication-based mirrored GPFS configuration with a tiebreaker site

Setting up an active-active GPFS configuration:

This example demonstrates how to configure an active-active GPFS cluster.

To establish an active-active GPFS cluster using hardware replication with a tiebreaker site as shown in Figure 14, consider the configuration:

Site A (production site)
> Consists of:
> - Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**
> - Storage subsystems – **A**
> - Disk volumes – **diskA** on storage system **A**
>   **diskA** is SAN-attached and accessible from sites **A** and **B**

Site B (recovery site)
> Consists of:
> - Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**

- Storage subsystems – **B**
- Disk volumes – **diskB** on storage system **B**

    diskB is SAN-attached and accessible from site **B** only

Site C (tiebreaker)

    Consists of:

- Nodes – **nodeC**

    diskC is an NSD defined over the internal disk of the node **nodeC** and is directly accessible only from site **C**

1. Establish a hardware replication connectivity between the storage systems and then establish the synchronous replication volume pair between the source and target using the copy entire volume option. In this case, it would be **diskA–diskB**.

2. In order to protect the order of dependent writes that span multiple disk volumes, multiple storage systems, or both, the consistency group functionality of the storage system should be used with all GPFS devices in the same consistency group.

3. Create a GPFS cluster defining the primary cluster configuration server as nodes **nodeA001** at site **A**, the secondary cluster configuration server as **nodeB001** at site **B**, an equal number of quorum nodes at each site, including the tiebreaker node at site **C**, **nodeC**. To prevent the tiebreaker node from assuming the role of file system manager, define it as **client**. Define all other quorum nodes as **manager**. List the nodes in the cluster in the file **NodeDescFile**. The **NodeDescFile** file contains the node descriptors:

```
nodeA001:quourum-manager
nodeA002:quorum-manager
nodeA003:quorum-manager
nodeA004:client
nodeB001:quorum-manager
nodeB002:quorum-manager
nodeB003:quorum-manager
nodeB004:client
nodeC:quorum-client
```

    Issue this command:

```
mmcrcluster –N NodeDescFile –p nodeA001 –s nodeB001
```

4. Enable the **unmountOnDiskFail** option on the tiebreaker node preventing false disk errors in the SAN configuration from being reported to the file system manager by issuing the **mmchconfig** command:

```
mmchconfig unmountOnDiskFail=yes -N nodeC
```

5. Create an NSD over **diskA**. The disk descriptor contained in the file **DiskDescFile** is:

```
/dev/diskA:nodeA001:nodeA002:dataAndMetadata:1
```

    Issue this command:

```
mmcrnsd –F DiskDescFileP
```

6. Start the GPFS daemon on all nodes:

```
mmstartup -a
```

7. Create a GPFS file system and mount it on all nodes at sites **A** and **B**.

```
mmcrfs /gpfs/fs0 fs0 -F DiskDescFile
```

Failover to the recovery site and subsequent failback for an active/active configuration:

For an active/active storage replication based cluster, complete these steps to restore access to the file system through site **B** after site **A** has experienced a disastrous failure.

Procedure when the server-based configuration scheme is in use

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file **gpfs.siteB** lists all of the nodes at site **B**:

   ```
   mmdsh -N gpfs.siteB /usr/lpp/mmfs/bin/mmshutdown
   ```

2. Perform the appropriate commands to make the secondary replication devices available and change their status from being secondary devices to suspended primary devices.

3. If you needed to relax node quorum or make configuration changes, migrate the primary cluster configuration server to site **B**, issue this command:

   ```
   mmchcluster -p nodeB001
   ```

4. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to non-quorum nodes:

   ```
   mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```

5. Ensure the source volumes are *not* accessible to the recovery site:
   - Disconnect the cable
   - Define the **nsddevices** user exit file to exclude the source volumes

6. Restart the GPFS daemon on all surviving nodes:

   ```
   mmstartup -N gpfs.siteB
   ```

Procedure when the Clustered Configuration Repository (CCR) scheme is in use

For an active-active PPRC-based cluster, follow these steps to restore access to the file system through site **B** after site **A** has experienced a disastrous failure:

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file gpfs.siteB lists all of the nodes at site **B**:

   ```
   mmshutdown -N gpfs.siteB
   ```

2. Perform the appropriate commands to make the secondary replication devices available and change their status from being secondary devices to suspended primary devices.

3. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to nonquorum nodes using the –- force option:

   ```
   mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC --force
   ```

4. Ensure that the source volumes are not accessible to the recovery site:
   - Disconnect the cable.
   - Define the **nsddevices** user exit file to exclude the source volumes.

5. Restart the GPFS daemon on all surviving nodes:

   ```
   mmstartup -N gpfs.siteB
   ```

Note:
- Make no further changes to the quorum designations at site B until the failed sites are back on line and the following failback procedure has been completed.
- Do not shut down the current set of nodes on the surviving site B and restart operations on the failed sites A and C. This will result in a non-working cluster.

Failback procedure

After the operation of site **A** has been restored, the failback procedure is completed to restore the access to the file system from that location. The following procedure is the same for both configuration schemes (server-based and Clustered Configuration Repository (CCR)). The failback operation is a two-step process:

1. For each of the paired volumes, resynchronize the pairs in the reserve direction with the recovery LUN **diskB** acting as the sources for the production LUN **diskA**. An incremental resynchronization is performed, which identifies the mismatching disk tracks, whose content is then copied from the recovery LUN to the production LUN. Once the data has been copied and the replication is running in the reverse direction this configuration can be maintained until a time is chosen to switch back to site **A**.

2. Shut GPFS down at site **B** and reverse the disk roles (the original primary disk becomes the primary again), bringing the replication pair to its initial state.

   a. Stop the GPFS daemon on all nodes.

   b. Perform the appropriate actions to switch the replication direction so that **diskA** is now the source and **diskB** is the target.

   c. If during failover you migrated the primary cluster configuration server to a node in site **B**:

      1) Migrate the primary cluster configuration server back to site **A**:

         ```
         mmchcluster -p nodeA001
         ```

      2) Restore the initial quorum assignments:

         ```
         mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
         ```

      3) Ensure that all nodes have the latest copy of the **mmsdrfs** file:

         ```
         mmchcluster -p LATEST
         ```

   d. Ensure the source volumes *are* accessible to the recovery site:

      - Reconnect the cable

      - Edit the **nsddevices** user exit file to *include* the source volumes

   e. Start the GPFS daemon on all nodes:

      ```
      mmstartup -a
      ```

   f. Mount the file system on all the nodes at sites **A** and **B**.

An active-passive GPFS cluster

In an active-passive environment, two GPFS clusters are set up in two geographically distinct locations (the production and the recovery sites). These clusters are referred to as peer GPFS clusters.

A GPFS file system is defined over a set of disk volumes located at the production site and these disks are mirrored using storage replication to a secondary set of volumes located at the recovery site. During normal operation, only the nodes in the production GPFS cluster mount and access the GPFS file system at any given time, which is the primary difference between a configuration of this type and the active-active model.

In the event of a catastrophe in the production cluster, the storage replication target devices are made available to be used by the nodes in the recovery site.

The secondary replica is then mounted on nodes in the recovery cluster as a regular GPFS file system, thus allowing the processing of data to resume at the recovery site. At a latter point, after restoring the physical operation of the production site, we execute the failback procedure to resynchronize the content of the replicated volume pairs between the two clusters and re-enable access to the file system in the production environment.

The high-level organization of synchronous active-passive storage replication based GPFS cluster is shown in Figure 15 on page 448.
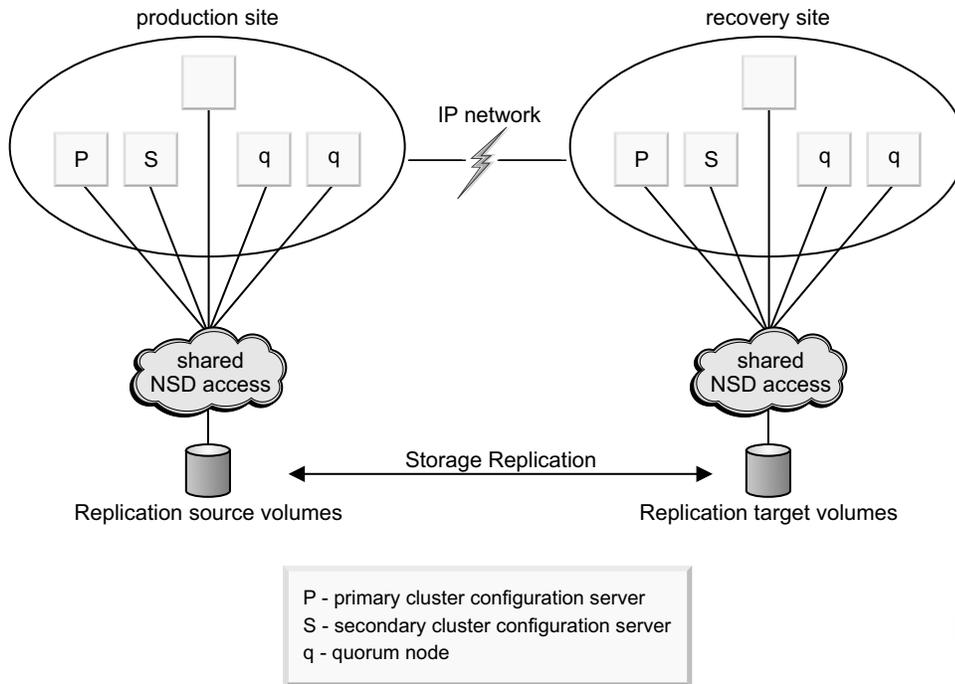
Figure 15. A synchronous active-passive storage replication-based GPFS configuration without a tiebreaker site

Setting up an active-passive GPFS configuration:

This example demonstrates how to configure an active-passive GPFS cluster.

To establish an active-passive storage replication GPFS cluster as shown in Figure 15, consider the configuration:

Production site
>   Consists of:
>   - Nodes – **nodeP001**, **nodeP002**, **nodeP003**, **nodeP004**, **nodeP005**
>   - Storage subsystems – Storage System **P**
>   - LUN IDs and disk volume names – **lunP1** (**hdisk11**), **lunP2** (**hdisk12**), **lunP3** (**hdisk13**), **lunP4** (**hdisk14**)

Recovery site
>   Consists of:
>   - Nodes – **nodeR001**, **nodeR002**, **nodeR003**, **nodeR004**, **nodeR005**
>   - Storage subsystems – Storage System **R**
>   - LUN ids and disk volume names – **lunR1** (**hdisk11**), **lunR2** (**hdisk12**), **lunR3** (**hdisk13**), **lunR4** (**hdisk14**)

All disks are SAN-attached and directly accessible from all local nodes.

1. Establish synchronous PPRC volume pairs using the **copy entire volume** option:

```
lunP1-lunR1 (source-target)
lunP2-lunR2 (source-target)
lunP3-lunR3 (source-target)
lunP4-lunR4 (source-target)
```

.

2. Create the recovery cluster selecting **nodeR001** as the primary cluster data server node, **nodeR002** as the secondary cluster data server nodes, and the nodes in the cluster contained in the file **NodeDescFileR**. The **NodeDescFileR** file contains the node descriptors:

```
nodeR001:quourum-manager
nodeR002:quourum-manager
nodeR003:quourum-manager
nodeR004:quourum-manager
nodeR005
```

Issue this command:

```
mmcrcluster –N NodeDescFileR –p nodeR001 –s nodeR002
```

3. Create the GPFS production cluster selecting **nodeP001** as the primary cluster data server node, **nodeP002** as the secondary cluster data server node, and the nodes in the cluster contained in the file **NodeDescFileP**. The **NodeDescFileP** file contains the node descriptors:

```
nodeP001:quourum-manager
nodeP002:quourum-manager
nodeP003:quourum-manager
nodeP004:quourum-manager
nodeP005
```

Issue this command:

```
mmcrcluster –N NodeDescFileP –p nodeP001 –s nodeP002
```

4. At all times the peer clusters must see a consistent image of the mirrored file system's configuration state contained in the **mmsdrfs** file. After the initial creation of the file system, all subsequent updates to the local configuration data must be propagated and imported into the peer cluster. Execute the **mmfsctl syncFSconfig** command to resynchronize the configuration state between the peer clusters after each of these actions in the primary GPFS cluster:

- Addition of disks through the **mmadddisk** command
- Removal of disks through the **mmdeldisk** command
- Replacement of disks through the **mmrpldisk** command
- Modifications to disk attributes through the **mmchdisk** command
- Changes to the file system's mount point through the **mmchfs -T** command

To automate the propagation of the configuration state to the recovery cluster, activate and use the **syncFSconfig** user exit. Follow the instructions in the prolog of **/usr/lpp/mmfs/samples/syncfsconfig.sample**.

5. From a node in the production cluster, start the GPFS daemon on all nodes:

```
mmstartup -a
```

6. Create the NSDs at the production site. The disk descriptors contained in the file **DiskDescFileP** are:

```
/dev/hdisk11:nodeP001:nodeP002:dataAndMetadata:-1
/dev/hdisk12:nodeP001:nodeP002:dataAndMetadata:-1
/dev/hdisk13:nodeP001:nodeP002:dataAndMetadata:-1
/dev/hdisk14:nodeP001:nodeP002:dataAndMetadata:-1
```

Issue this command:

```
mmcrnsd –F DiskDescFileP
```

7. Create the GPFS file system and mount it on all nodes at the production site:

```
mmcrfs /gpfs/fs0 fs0 -F DiskDescFileP
```

Failover to the recovery site and subsequent failback for an active-passive configuration:

For an active-passive storage replication based cluster, complete these steps to fail over production to the recovery site.

Procedure when the server-based configuration scheme is in use

1. If the GPFS daemon is not already stopped on all surviving nodes in the production cluster, from a node in the production cluster issue:

   `mmshutdown –a`

2. Perform the appropriate commands to make the secondary replication devices available and change their status from being secondary devices to suspended primary devices.

3. From a node in the recovery cluster start GPFS:

   `mmstartup –a`

4. Mount the file system on all nodes in the recovery cluster.

Procedure when the Clustered Configuration Repository (CCR) scheme is in use

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file `gpfs.siteB` lists all of the nodes at site **B**:

   `mmshutdown -N gpfs.siteB`

2. Perform the appropriate commands to make the secondary replication devices available and change their status from being secondary devices to suspended primary devices.

3. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to nonquorum nodes using the –- force option:

   `mmchnode --nonquorum -N nodeA001, nodeA002, nodeA003, nodeC --force`

4. Ensure that the source volumes are *not* accessible to the recovery site:

   - Disconnect the cable
   - Define the **nsddevices** user exit file to exclude the source volumes

5. Restart the GPFS daemon on all surviving nodes:

   `mmstartup -N gpfs.siteB`

Note: Make no further changes to the quorum designations at site B until the failed sites are back on line and the following failback procedure has been completed. Do not shut down the current set of nodes on the surviving site B and restart operations on the failed sites A and C. This will result in a non-working cluster.

Failback procedure

After the physical operation of the production site has been restored, complete the failback procedure to transfer the file system activity back to the production GPFS cluster. The following procedure is the same for both configuration schemes (server-based and Clustered Configuration Repository (CCR)). The failback operation is a two-step process:

1. For each of the paired volumes, resynchronize the pairs in the reserve direction with the recovery LUN **lunRx** acting as the sources for the production LUN **lunPx**. An incremental resynchronization will be performed which identifies the mismatching disk tracks, whose content is then copied from the recovery LUN to the production LUN. Once the data has been copied and the replication is running in the reverse direction this configuration can be maintained until a time is chosen to switch back to site **P**.

2. If the state of the system configuration has changed, update the GPFS configuration data in the production cluster to propagate the changes made while in failover mode. From a node at the recovery site, issue:

   `mmfsctl all syncFSconfig –n gpfs.sitePnodes`

3. Stop GPFS on all nodes in the recovery cluster and reverse the disk roles so the original primary disks become the primaries again:

   a. From a node in the recovery cluster, stop the GPFS daemon on all nodes in the recovery cluster:

```
mmshutdown -a
```

b. Perform the appropriate actions to switch the replication direction so that **diskA** is now the source and **diskB** is the target.

c. From a node in the production cluster, start GPFS:

```
mmstartup -a
```

d. From a node in the production cluster, mount the file system on all nodes in the production cluster.

Point In Time Copy of IBM Spectrum Scale data

Most storage systems provide the functionality to make a point-in-time copy of data as an online backup mechanism. This function provides an instantaneous copy of the original data on the target disk, while the actual copy of data takes place asynchronously and is fully transparent to the user.

Several uses of the FlashCopy replica after its initial creation can be considered. For example, if your primary operating environment suffers a permanent loss or a corruption of data, you may choose to flash the target disks back onto the originals to quickly restore access to a copy of the file system as seen at the time of the previous snapshot. Before restoring the file system from a FlashCopy, please make sure to suspend the activity of the GPFS client processes and unmount the file system on all GPFS nodes. FlashCopies also can be used to create a copy of data for disaster recovery testing and in this case are often taken from the secondary devices of a replication pair.

When a FlashCopy disk is first created, the subsystem establishes a control bitmap that is subsequently used to track the changes between the source and the target disks. When processing read I/O requests sent to the target disk, this bitmap is consulted to determine whether the request can be satisfied using the target's copy of the requested block. If the track containing the requested data has not yet been copied, the source disk is instead accessed and its copy of the data is used to satisfy the request.

To prevent the appearance of out-of-order updates, it is important to consider data consistency when using FlashCopy. When taking the FlashCopy image all disk volumes that make up the file system must be copied so that they reflect the same logical point in time. Two methods may be used to provide for data consistency in the FlashCopy image of your GPFS file system. Both techniques guarantee the consistency of the FlashCopy image by the means of temporary suspension of I/O, but either can be seen as the preferred method depending on your specific requirements and the nature of your GPFS client application.

FlashCopy provides for the availability of the file system's on-disk content in another GPFS cluster. But in order to make the file system known and accessible, you must issue the **mmfsctl syncFSConfig** command to:

- Import the state of the file system's configuration from the primary location.
- Propagate all relevant changes to the configuration in the primary cluster to its peer to prevent the risks of discrepancy between the peer's **mmsdrfs** file and the content of the file system descriptor found in the snapshot.

  It is suggested you generate a new FlashCopy replica immediately after every administrative change to the state of the file system. This eliminates the risk of a discrepancy between the GPFS configuration data contained in the **mmsdrfs** file and the on-disk content of the replica.

Using consistency groups for Point in Time Copy

This topic provides a description about using consistency groups for point in time copy mechanism in IBM Spectrum Scale.

The use of FlashCopy consistency groups provides for the proper ordering of updates, but this method does not by itself suffice to guarantee the atomicity of updates as seen from the point of view of the user application. If the application process is actively writing data to GPFS, the on-disk content of the file system may, at any point in time, contain some number of incomplete data record updates and possibly

some number of in-progress updates to the GPFS metadata. These appear as partial updates in the FlashCopy image of the file system, which must be dealt before enabling the image for normal file system use. The use of metadata logging techniques enables GPFS to detect and recover from these partial updates to the file system's metadata. However, ensuring the atomicity of updates to the actual data remains the responsibility of the user application. Consequently, the use of FlashCopy consistency groups is suitable only for applications that implement proper mechanisms for the recovery from incomplete updates to their data.

The FlashCopy consistency group mechanism is used to freeze the source disk volumes at the logical instant at which their logical image appears on the target disk volumes. The appropriate storage system documentation should be consulted to determine how to invoke the Point in Time Copy with the consistency group option:

Assuming a configuration with:
- Storage subsystems – 1
- LUN ids and disk volume names – **lunS1** (**hdisk11**), **lunS2** (**hdisk12**), **lunT1**, **lunT2**

  lunS1 and **lunS2** are the FlashCopy source volumes. These disks are SAN-connected and appear on the GPFS nodes as **hdisk11** and **hdisk12**, respectively. A single GPFS file system **fs0** has been defined over these two disks. **lunT1** and **lunT2** are the FlashCopy target volumes. None of the GPFS nodes have direct connectivity to these disks.

To generate a FlashCopy image using a consistency group, do the following step:

Run the **establish FlashCopy pair** task with the **freeze FlashCopy consistency group** option. Create the volume pairs:

```
lunS1 – lunT1  (source-target)
lunS2 – lunT2  (source-target)
```

Using file-system-level suspension for Point in Time Copy

The use of file-system-level suspension through the `mmfsctl` command prevents incomplete updates in the FlashCopy image and is the suggested method for protecting the integrity of your FlashCopy images. Issuing the `mmfsctl` command leaves the on-disk copy of the file system in a fully consistent state, ready to be flashed and copied onto a set of backup disks. The command instructs GPFS to flush the data buffers on all nodes, write the cached metadata structures to disk, and suspend the execution of all subsequent I/O requests.

1. To initiate file-system-level suspension, issue the **mmfsctl suspend** command.
2. To resume normal file system I/O, issue the **mmfsctl resume** command.

Assuming a configuration with:
- Storage subsystems – ESS 1; logical subsystem LSS 1
- LUN ids and disk volume names – **lunS1** (**hdisk11**), **lunS2** (**hdisk12**), **lunT1**, **lunT2**

  lunS1 and **lunS2** are the FlashCopy source volumes. These disks are SAN-connected and appear on the GPFS nodes as **hdisk11** and **hdisk12**, respectively. A single GPFS file system **fs0** has been defined over these two disks.

  lunT1 and **lunT2** are the FlashCopy target volumes. None of the GPFS nodes have direct connectivity to these disks.

To generate a FlashCopy image using file-system-level suspension:
1. From any node in the GPFS cluster, suspend all file system activity and flush the GPFS buffers on all nodes:

   ```
   mmfsctl fs0 suspend
   ```
2. Run the **establish FlashCopy pair** task to create the following volume pairs:

   ```
   lunS1 – lunT1  (source-target)
   lunS2 – lunT2  (source-target)
   ```

3. From any node in the GPFS cluster, resume the file system activity:

```
mmfsctl fs0 resume
```

Chapter 30. Implementing a clustered NFS environment on Linux

In addition to the traditional exporting of GPFS file systems using the Network File System (NFS) protocol, GPFS allows you to configure a subset of the nodes in the cluster to provide a highly-available solution for exporting GPFS file systems using NFS.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

The participating nodes are designated as Cluster NFS (CNFS) member nodes and the entire setup is frequently referred to as CNFS or a CNFS cluster.

In this solution, all CNFS nodes export the same file systems to the NFS clients. When one of the CNFS nodes fails, the NFS serving load moves from the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover. For the NFS client node to experience a seamless failover, hard mounts must be used. The use of soft mounts will likely result in stale NFS file handle conditions when a server experiences a problem, even though CNFS failover will still be done.

Currently, CNFS is supported only in the Linux environment. For an up-to-date list of supported operating systems, specific distributions, and other dependencies, refer to the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

NFS monitoring

Every node in the CNFS cluster runs a separate GPFS utility that monitors GPFS, NFS, and networking components on the node. Upon failure detection and based on your configuration, the monitoring utility might invoke a failover.

While an NFS server is in a grace period, the NFS monitor sets the server's NFS state to "Degraded".

NFS failover

As part of GPFS recovery, the CNFS cluster failover mechanism is invoked. It transfers the NFS serving load that was served by the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover.

The failover mechanism is based on IP address failover. The CNFS IP address is moved from the failing node to a healthy node in the CNFS cluster. In addition, it guarantees NFS lock (NLM) recovery.

Failover processing may involve rebooting of the problem node. To minimize the effects of the reboot, it is recommended that the CNFS nodes be dedicated to that purpose and are not used to run other critical processes. CNFS node rebooting should not be disabled or the failover reliability will be severely impacted.

NFS locking and load balancing

CNFS supports a failover of all of the node's load together (all of its NFS IP addresses) as one unit to another node. However, if no locks are outstanding, individual IP addresses can be moved to other nodes for load balancing purposes.

CNFS is dependent on DNS for any automated load balancing of NFS clients among the NFS cluster nodes. Using the round-robin algorithm is highly recommended.

CNFS network setup

In addition to one set of IP addresses for the GPFS cluster, a separate set of one or more IP addresses is required for CNFS serving.

The GPFS cluster can be defined over an IPv4 or IPv6 network. The IP addresses specified for CNFS can also be IPv4 or IPv6. The GPFS cluster and CNFS are not required to be on the same version of IP, but IPv6 must be enabled on GPFS to support IPv6 on CNFS.

CNFS setup

You can set up a clustered NFS environment within a GPFS cluster.

To do this, follow these steps:

1. Designate a separate directory for the CNFS shared files:

   mmchconfig cnfsSharedRoot=*directory*

   where:

   cnfsSharedRoot=*directory*
   > Is the path name to a GPFS directory, preferably on a small separate file system that is not exported by NFS. The GPFS file system that contains the directory must be configured to be mounted automatically upon GPFS start on each of the CNFS nodes (**-A yes** option on the **mmchfs** command). **cnfsSharedRoot** is a mandatory parameter and must be defined first.

2. Add all GPFS file systems that need to be exported to **/etc/exports**. For NSF export considerations, see the topic *Exporting a GPFS file system using NFS* in the *IBM Spectrum Scale: Administration Guide*.

3. If the shared directory from step 1 is in an exported file system, restrict access to that directory.

4. Use the **mmchnode** command to add nodes to the CNFS cluster:

   mmchnode --cnfs-interface=*ip_address_list* **-N** *node*

   where:

   ip_address_list
   > Is a comma-separated list of host names or IP addresses to be used for GPFS cluster NFS serving.

   node
   > Identifies a GPFS node to be added to the CNFS cluster.

   For more information, see the topic *mmchnode command* in the *IBM Spectrum Scale: Command and Programming Reference*.

5. Use the **mmchconfig** command to configure the optional CNFS parameters.

   cnfsMountdPort=*mountd_port*
   > Specifies the port number to be used for the **rpc.mountd** daemon.

   > For CNFS to work correctly with the automounter (AMD), the **rpc.mountd** daemon on the different nodes must be bound to the same port.

   cnfsNFSDprocs=*nfsd_procs*
   > Specifies the number of **nfsd** kernel threads. The default is 32.

   cnfsVersions=*nfs_versions*
   > Specifies a comma-separated list of protocol versions that CNFS should start and monitor. The default is **3,4**. If you are not using NFS v3 and NFS v4, specify this parameter with the appropriate values for your configuration.

Note: If you are not using NFS v3 and NFS v4, and you do not explicitly specify **cnfsVersions** with the protocol versions on your system, the following message will continually appear in the **mmfs.log**:

```
Found NFS version mismatch between CNFS and current running config, check the OS config files.
```

6. If multiple failover groups are desired, assign a group ID to each NFS node:

   mmchnode --cnfs-groupid=_groupid_ **-N** _node_

   To assign NFS nodes to different groups, use a group ID that is in a different range of ten. For example, a node with group ID 2n will fail over only to nodes in the same range of ten (which means any node with group ID 20 to 29). Failover in the same group will first look for one of the nodes with the same group ID. If none are found, any node in the group range starting at n0 to n9 is selected.

CNFS administration

There are some common CNFS administration tasks in this topic along with a sample configuration.

To query the current CNFS configuration, enter:

mmlscluster --cnfs

To temporarily disable CNFS on one or more nodes, enter:

mmchnode --cnfs-disable -N _NodeList_

Note: This operation affects only the high-availability aspects of the CNFS functionality. Normal NFS exporting of the data from the node is not affected. All currently defined CNFS IP addresses remain unchanged. There will be no automatic failover from or to this node in case of a failure. If failover is desired, GPFS should be shut down on the affected node prior to issuing the **mmchnode** command.

To re-enable previously-disabled CNFS member nodes, enter:

mmchnode --cnfs-enable -N _NodeList_

Note: If the GPFS daemon is running on a node on which CNFS is being re-enabled, the node will try to activate its CNFS IP address. If the IP address was currently on some other CNFS-enabled node, that activation would include a takeover.

To permanently remove nodes from the CNFS cluster, enter:

mmchnode --cnfs-interface=DELETE -N _NodeList_

Note: This operation affects only the high-availability aspects of the CNFS functionality. Normal NFS exporting of the data from the node is not affected. All currently defined CNFS IP addresses remain unchanged. There will be no automatic failover from or to this node in case of a failure. If failover is desired, GPFS should be shut down on the affected node prior to issuing the **mmchnode** command.

A sample CNFS configuration

Here is a CNFS configuration example, which assumes the following:
- Your GPFS cluster contains three nodes: `fin18`, `fin19`, and `fin20`
- The host names for NFS serving are: `fin18nfs`, `fin19nfs`, and `fin20nfs`

To define a CNFS cluster made up of these nodes, follow these steps:
1. Add the desired GPFS file systems to **/etc/exports** on each of the nodes.
2. Create a directory called **ha** in one of the GPFS file systems by entering:
   ```
   mkdir /gpfs/fs1/ha
   ```
3. Create a temporary file called **/tmp/hanfs-list**, which contains the following lines:

```
fin18 --cnfs-interface=fin18nfs
fin19 --cnfs-interface=fin19nfs
fin20 --cnfs-interface=fin20nfs
```

4. Set the CNFS shared directory by entering:

   `mmchconfig cnfsSharedRoot=/gpfs/fs1/ha`

5. Create the CNFS cluster with the **mmchnode** command, by entering:

   `mmchnode -S /tmp/hanfs-list`

6. Access the exported GPFS file systems over NFS. If one or more GPFS nodes fail, the NFS clients should continue uninterrupted.

Chapter 31. Implementing Cluster Export Services

Cluster Export Services (CES) provides highly available file and object services to a GPFS cluster by using Network File System (NFS), Object, or Server Message Block (SMB) protocols.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

CES is an alternate approach to using a clustered Network File System (CNFS) to export GPFS file systems. For more information about CES and protocol configuration, see Chapter 2, "Configuring the CES and protocol configuration," on page 25.

CES features

To successfully use Cluster Export Services (CES), you must consider function prerequisites, setup and configuration, failover/failback policies, and other management and administration requirements.

CES cluster setup

You can set up a Cluster Export Services (CES) environment within a GPFS cluster.

The CES shared root (cesSharedRoot) directory is needed for storing CES shared configuration data, protocol recovery, and some other protocol-specific purposes. It is part of the Cluster Export Configuration and is shared between the protocols. Every CES node requires access to the path that is configured as shared root.

To update the CES shared root directory, you must shut down the cluster, set the CES shared root directory, and start the cluster again:
```
mmshutdown -a
mmchconfig cesSharedRoot=shared_root_path
mmstartup -a
```

The recommendation for CES shared root directory is a dedicated file system. It can also reside in an existing GPFS file system. In any case, the CES shared root directory must be on GPFS and must be available when it is configured through the **mmchconfig** command.

To enable protocol nodes, the CES shared root directory must be defined. To enable protocol nodes, use the following command:
```
mmchnode --ces-enable -N Node1[,Node2...]
```

To disable a CES node, use the following command:
```
mmchnode --ces-disable -N Node1[,Node2...]
```

Preparing to perform service actions on the CES shared root directory file system

The CES shared root directory file system must be kept available for protocols operation to function. If a service action is to be performed on the CES shared root directory file system, perform the steps that follow.

Commands such as **mmshutdown, mmstartup and mmmount**, can be passed in the `cesnodes` node class parameter to ensure operation on all protocol nodes.

The following steps are used to perform service actions on the CES shared root file system:

1. Inform users of the impact to protocols. Quiesce protocol related I/O and mounts if possible. Quiesce cluster functions in progress on protocol nodes such as recalls, migrations, AFM, backup, and any policies that may be in use on the protocol nodes; or transition these cluster functions to other nodes. Finally, verify that file system quorum can be achieved by the remaining cluster nodes.

2. Shut down GPFS on all protocol nodes:

   ```
   mmshutdown -N cesnodes
   ```

   Note: Only protocol nodes need to be shut down for service of the CES shared root directory file system. However, other nodes may need to unmount the file system, depending on what service is being performed.

Protocol nodes are now ready for service actions to be performed on the CES shared root directory or the nodes themselves. To recover from a service action:

1. Start up GPFS on all protocol nodes:

   ```
   mmstartup -N cesnodes
   ```

2. Make sure that the CES shared root directory file system is mounted on all protocol nodes:

   ```
   mmmount cesSharedRoot -N cesnodes
   ```

3. Verify that all protocol services have been started:

   ```
   mmces service list -a
   ```

CES network configuration

Cluster Export Services (CES) IP addresses are used to export data via the NFS, SMB, and Object protocols. File and Object clients use the public IPs to access data on GPFS file systems.

CES IP addresses have the following characteristics:
- Shared between all CES protocols
- Organized in an *address pool* (there can be fewer or more CES addresses than nodes)
- Hosted on the CES nodes (there can be CES nodes without CES addresses)
- Can move between CES nodes (triggered manually via the command or as part of a CES failover)
- Must not be used for GPFS communication at the same time

CES IP addresses have these restrictions:
- The network on CES nodes must be configured so that all CES IPs can run on any CES node. Typically this configuration requires that all CES nodes have at least one NIC interface or VLAN-compatible interface with each CES IP network address. If different subnets are used, then all the CES IPs in a given CES group must be able to run on any node in that group.
- CES IPs are created as aliases on each CES node. Do not include the primary address of an adapter in the CES IP address pool.
- CES IPs must be resolvable by DNS or /etc/hosts.
- CES does not manage the subnet or netmask configuration. It is the user's task.

To add CES IP addresses to the address pool, use the **mmces** command:

```
mmces address add --ces-ip Address[,Address...]
```

By default, addresses are distributed among the CES nodes, but a new address can be assigned to a particular node:

```
mmces address add --ces-ip Address[,Address...] --ces-node Node
```

After a CES IP address is added to the address pool, you can manually move the address to a particular node:

```
mmces address move --ces-ip Address[,Address...] --ces-node Node
```

You can remove a CES IP address from the address pool with the **mmces** command:

```
mmces address remove --ces-ip Address[,Address...]
```

Removing an address while there are clients connected causes the clients to lose those connections. Any reconnection to the removed IP results in a failure. If DNS is used to map a name entry to one or more IP addresses, update the DNS to ensure that a client is not presented an address that was already removed from the pool. This process might also include invalidation of any DNS caches.

CES addresses are virtual IP addresses

The CES addresses that are assigned to the CES nodes are implemented as IP aliases. Each network adapter that hosts CES addresses must already be configured (with different non-CES IPs) in `/etc/sysconfig`. CES uses the netmask to figure out which interfaces to use. For example, if `eth1` is 10.1.1.1 and `eth2` is 9.1.1.1, then the CES IP 10.1.1.100 maps to `eth1` and the CES IP 9.1.1.100 maps to `eth2`.

Virtual IP addresses include the following advantages:

- The node does not need to wait for the switch to accept the link when an IP address is failed back. Since the address is an alias, the interface on which it resides is already up.
- IP address failover is much faster.
- Administration is simplified by providing a clear distinction between the *system IP* and the *CES IP*. For example, you have a two-node cluster. One of the nodes in the two-node cluster has a problem that induces failover and someone logs in to the suspected node to reboot it. The surviving node might get rebooted by accident if the system address was migrated to the surviving node.

How to use an alias

To use an alias address for CES, you need to provide a static IP address that is not already defined as an alias in the `/etc/sysconfig/network-scripts` directory.

Before you enable the node as a CES node, configure the network adapters for each subnet that are represented in the CES address pool:

1. Define a static IP address for the device:

```
/etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=none
IPADDR=10.1.1.10
NETMASK=255.255.255.0
ONBOOT=yes
GATEWAY=10.1.1.1
TYPE=Ethernet
```

2. Ensure that there are no aliases that are defined in the network-scripts directory for this interface:

```
# ls -l /etc/sysconfig/network-scripts/ifcfg-eth1:*
ls: /etc/sysconfig/network-scripts/ifcfg-eth1:*: No such file or directory
```

After the node is enabled as a CES node, no further action is required. CES addresses are added as aliases to the already configured adapters.

CES address failover and distribution policies

When a Cluster Export Services (CES) node leaves the GPFS cluster, any CES IP addresses that are assigned to that node are moved to CES nodes still within the cluster. Additionally, certain error conditions and administrative operations can cause a node to release its addresses to be reassigned to other nodes.

As CES nodes enter and leave the GPFS cluster, the addresses are distributed among the nodes according to the address distribution policy that is selected. In addition, you can disable automatic address distribution to allow the user to manually maintain the address-to-node assignments.

The address distribution policy is set with the **mmces** command:

```
mmces address policy [even-coverage | balanced-load | node-affinity | none]
```

The following list describes each type of address distribution policy:

even-coverage
> Distributes the addresses among the available nodes. The even-coverage policy is the default address distribution policy.

balanced-load
> Distributes the addresses to approach an optimized load distribution. The load (network and CPU) on all the nodes are monitored. Addresses are moved based on given policies for optimized load throughout the cluster.

node-affinity
> Attempts to keep an address on the node to which the user manually assigned it. If the **mmces address add** command is used with the `--ces-node` option, the address is marked as being associated with that node. Similarly, if an address is moved with the **mmces address move** command, the address is marked as being associated with the destination node. Any automatic movement, such as reassigning a down node's addresses, does not change this association. Addresses that are enabled with no node specification do not have a node association.
>
> Addresses that are associated with a node but assigned to a different node are moved back to the associated node if possible.

Automatic address distribution is performed in the background in a way as to not disrupt the protocol servers more than necessary. If you want immediate redistribution of the addresses, use the **mmces** command to force an immediate rebalance:

```
mmces address move --rebalance
```

In order to prevent an interruption in service, IP addresses that have attributes assigned to them (for example: **object_database_node** or **object_singleton_node**) are not rebalanced.

You can further control the assignment of CES addresses by placing nodes or addresses in CES groups. For more information, see the topic "Configuring CES protocol service IP addresses" on page 26.

CES protocol management

Cluster Export Services (CES) protocols (NFS, SMB, and Object) are enabled or disabled with the **mmces** command.

Command examples:

```
mmces service enable [NFS | OBJ | SMB]
mmces service disable [NFS | OBJ | SMB]
```

After a protocol is enabled, the protocol is started on all CES nodes.

When a protocol is disabled, the protocol is stopped on all CES nodes and all protocol-specific configuration data is removed.

CES management and administration

Cluster Export Services (CES) nodes can be suspended for maintenance reasons with the **mmces** command.

For example:

```
mmces node suspend [-N Node[,Node...]
```

When a node is suspended:

- The GPFS state of the node is unaffected. It remains an active member of the cluster.
- All CES IP addresses that are assigned to the node are reassigned to other nodes. Assignment of addresses to a suspend node is not allowed.
- All CES monitoring operations are stopped.
- Servers for the enabled CES protocols continue to run, but can be stopped.
- Suspended protocol servers can be stopped and started on the node with the following **mmces** commands.

```
mmces service stop [NFS | OBJ | SMB] [-N Node[,Node...]]
mmces service start [NFS | OBJ | SMB] [-N Node[,Node...]]
```

- A node or a group of nodes can be suspended and resume normal operation with the following **mmces** commands.

```
mmces node suspend -N node1,node2,node3
mmces node resume
```

After a node is resumed, monitoring on the node is started and the node is eligible for address assignments.

CES NFS support

In Cluster Export Services (CES), you must consider supported protocol versions, service and export configuration, NFS service monitoring, fail-over considerations, and client support requirements for Network File System (NFS) support.

NFS support levels

NFS versions 3 (NFSv3) and 4 (NFSv4.0) are supported.

NFS monitoring

The NFS servers are monitored to check for proper functions. If a problem is found, the CES addresses of the node are reassigned, and the node state is set to `failed`. When the problem is corrected, the node resumes normal operation.

NFS service configuration

Configuration options for the NFS service can be set with the **mmnfs config** command.

You can use the **mmnfs config** command to set and list default settings for NFS such as the port number for the NFS service, the default access mode for exported file systems, the log level, and enable or disable status for delegations. For a list of configurable attributes, see the topic *mmnfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Some of the attributes such as the protocol can be overruled for a given export on a per-client base. For example, the default settings might have NFS protocols 3 and 4 enabled, but the export for a client might restrict it to NFS version 4 only.

NFS export configuration

Exports can be added, removed, or changed with the **mmnfs export** command. Authentication must be set up before you define an export.

Exports can be declared for any directory in the GPFS file system, including a fileset junction. At the time where exports are declared, these folders must exist physically in GPFS. Only folders in the GPFS file system can be exported. No folders that are located only locally on a server node can be exported because they cannot be used in a failover situation.

Export-add and export-remove operations can be applied at run time of the NFS service. The export-change operation does require a restart of the NFS service on all server nodes that is followed by a 60-second grace period to allow connected clients to reclaim their locks and to avoid concurrent lock requests from new clients.

NFS failover

When a CES node leaves the cluster, the CES addresses assigned to that node are redistributed among the remaining nodes. Remote clients that access the GPFS file system might see a pause in service while the internal state information is passed to the new servers.

NFS clients

When you work with NFS clients, consider the following points:
- If you mount the same NFS export on one client from two different IBM Spectrum Scale NFS protocol nodes, data corruption might occur.
- Cross-protocol notifications are not supported by clustered NFS. For example, files that are created with NFS are not automatically visible on SMB clients and SMB clients need to refresh the directory listing by performing a manual refresh in Microsoft Windows Explorer. Similarly, files that are created from other methods such as files that are created with FTP or files that are restored from a backup are not automatically visible.
- The NFS protocol version that is used as the default on a client operating system might differ from what you expect. If you are using a client that mounts NFSv3 by default, and you want to mount NFSv4, then you must explicitly specify NFSv4 in the `mount` command. For more information, see the `mount` command for your client operating system.
- To prevent NFS clients from encountering data integrity issues during failover, ensure that NFS clients are mounted with the options `-o sync` and `-o hard`.
- A client must mount an NFS export by using the IP address of the GPFS system. If a host name is used, ensure that the name is unique and remains unique.

  If a DNS Round Robin (RR) entry name is used to mount an NFSv3 export, data unavailability might occur, due to unreleased locks. The lock manager on the GPFS file system is not *clustered-system-aware*.
- Reusing an export ID might cause unexpected behavior on a running system. When an export is removed, avoid reusing the export ID either for a different export or for reexporting the same file system.

Coexistence with clustered NFS and migration

CES and clustered NFS cannot exist in the same GPFS cluster.

Choosing between CNFS or CES

If you want to put highly available NFS services on top of the GPFS file system, you have the choice between clustered NFS (Chapter 30, "Implementing a clustered NFS environment on Linux," on page 455) and Cluster Export Services (Chapter 31, "Implementing Cluster Export Services," on page 459).

To help you choose one of these NFS offerings, consider the following points:

Multiprotocol support
     If you plan to use other protocols (such as SMB or Object) in addition to NFS, CES must be

chosen. While CNFS provides support only for NFS, the CES infrastructure adds support also for SMB and Object. With CES, you can start with NFS and add (or remove) other protocols at any time.

Command support

While CNFS provides native GPFS command support for creation and management of the CNFS cluster, it lacks commands to manage the NFS service and NFS exports. The CES infrastructure introduces native GPFS commands to manage the CES cluster. Furthermore, there are also commands to manage the supported protocol services and the NFS exports. For example, with CES, you do not need to adapt NFS configuration files individually on the protocol nodes. This work is done by the new GPFS commands that are provided for CES.

Performance

CNFS is based on the kernel NFS server while NFS support in CES is based on the Ganesha NFS server operating in user space. Due to the different nature of these NFS I/O stacks, performance depends on system characteristics and NFS workload. Contact your IBM representative to get help with sizing the required number of protocol nodes to support certain workload characteristics and protocol connection limits.

There is no general answer about which of the two NFS servers performs better as this depends on many factors. Tests that are conducted with both NFS I/O stacks over various workloads show that the kernel-based NFS server (CNFS) performs better under metadata-intensive workloads. Typically this testing is with many smaller files and structures. The Ganesha NFS server provides better performance on other data-intensive workloads such as video streaming.

Note: CES provides a different interface to obtain performance metrics for NFS. CNFS uses the existing interfaces to obtain NFS metrics from the kernel (such as `nfsstat` or the `/proc` interface). The CES framework provides the **`mmperfmon query`** command for Ganesha-based NFS statistics. For more information, see the topic *mmperfmon command* in the *IBM Spectrum Scale: Command and Programming Reference*.

SELinux support

CES, including the CES framework as well as SMB and CES NFS, does not support SELinux in enforcing mode.

Migration of CNFS to CES

For information about migrating existing CNFS environments to CES, see "Migration of CNFS clusters to CES clusters" on page 469.

CES SMB support

In GPFS 4.1.1 and later, you can access a GPFS file system with an SMB client using its inherent SMB semantics.

The following features are provided:

Note: Some of the features described below require a higher version than 4.1.1.

Clustered SMB support

SMB clients can connect to any of the protocol nodes and get access to the shares defined. A clustered registry makes sure that all nodes see the same configuration data. Therefore, clients can connect to any Cluster Export Services (CES) node and see the same data. Moreover, the state of opened files (share modes, open modes, access masks, locks, and so on) is also shared among the CES nodes so that data integrity is maintained. On failures, clients can reconnect to another protocol node and IP addresses are transferred to another protocol node

The supported protocol levels are SMB2 and the base functions of SMB3 (dialect negotiation, secure negotiation, encryption of data on the wire).

Export management command

With the `mmsmb` command, IBM Spectrum Scale provides a comprehensive entry point to manage all SMB-related configuration tasks like creating, changing, and deleting SMB shares.

SMB monitoring

The monitoring framework detects issue with the SMB services and triggers failover in case of an unrecoverable error.

Integrated installation

The SMB services are installed by the integrated installer together with the CES framework and the other protocols NFS and Object.

SMB performance metrics

The SMB services provide two sets of performance metrics that are collected by the performance monitor framework. Both current and historic data (with lower granularity) can be retrieved. The two sets of metrics are global SMB metrics (such as the number of connects and disconnects) and metrics for each SMB request (number, time, throughput). The `mmperfmon` query tool provides access to the most important SMB metrics via predefined queries. Moreover, metrics for the clustered file metadata database CTDB are collected and exposed via the `mmperfmon query` command.

Authentication and mapping

The SMB services can be configured to authenticate against the authentication services Microsoft Active Directory and LDAP. Mapping Microsoft security identifiers (SIDs) to the POSIX user and group IDs on the file server can either be done automatically by using the so-called autorid mechanism or external mapping services like RFC 2307 or Microsoft Services for Unix. If none of the offered authentication and mapping schemes matches the environmental requirements, a user-defined configuration can be established.

CES OBJ support

In Cluster Export Services (CES), you must consider several types of requirements for Object (OBJ) support.

Openstack support levels

The Mitaka release of OpenStack is used for Swift, Keystone, and their dependent packages.

The Swift V1 and Keystone V2 and V3 APIs are also supported.

Object monitoring

The object servers are monitored to ensure that they function properly. If a problem is found, the CES addresses of the node are reassigned, and the node state is set to failed. When the problem is corrected, the node resumes normal operation.

Object service configuration

The Object service configuration is controlled by the respective Swift and Keystone configuration files. The master versions of these files are stored in the CCR repository, and copies exist in the `/etc/swift` and `/etc/keystone` directories on each protocol node. The files that are stored in those directories should not be directly modified since they are overwritten by the files that are stored in the CCR. To change the Swift or Keystone configuration, use the `mmobj config change` command to modify the master copy of configuration files stored in CCR. The monitoring framework is notified of the change and propagates the file to the local file system of the CES nodes. For information about the values that can be changed and their associated function, refer to the administration guides for Swift and Keystone.

To change the authentication that is used by the Keystone server, use the `mmuserauth` command to change the authentication repository to AD or LDAP, or to enable SSL communication to the Keystone server.

Object fileset configuration

A base fileset must be specified when the Object service is configured. An existing fileset can be used or a new fileset can be created. All filesets are created in the GPFS file system that is specified during installation. This fileset is automatically created in the GPFS file system that is specified during installation. Evaluate the data that is expected to be stored by the Object service to determine the required number of inodes that are needed. This expected number of inodes is specified during installation, but can be updated later by using standard GPFS file system and fileset management commands.

Object failover

When a CES node leaves the cluster, the CES addresses that are assigned to that node are redistributed among the remaining nodes. Remote clients that access the Object service might see active connections drop or a pause in service while the while the CES addresses are moved to the new servers. Clients with active connections to the CES addresses that are migrated might have their connections unexpectedly drop. Clients are expected to retry their requests when this happens.

Certain Object-related services can be migrated when a node is taken offline. If the node was hosting the backend database for Keystone or certain Swift services that are designated as singletons (such as the auditor), those services are started on the active node that received the associated CES addresses of the failed node. Normal operation of the Object service resumes after the CES addresses are reassigned and necessary services automatically restarted.

Object clients

The Object service is based on Swift and Keystone, and externalizes their associated interfaces. Clients should follow the associated specifications for those interfaces. Clients must be able to handle dropped connections or delays during CES node failover. In such situations, clients should retry the request or allow more time for the request to complete.

To connect to an Object service, clients should use a load balancer or DNS service to distribute requests among the pool of CES IP addresses. Clients in a production environment should not use hard-coded CES addresses to connect to Object services. For example, the authentication URL should refer to a DNS host name or a load balancer front end name such as `http://protocols.gpfs.net:35357/v3` rather than a CES address.

Inode Allocation Overview

Object storage consumes fileset inodes when the unified file and object access layout is used. One inode is used for each file or object, and one inode is used for each directory in the object path.

In the traditional object layout, objects are placed in the following directory path:

gpfs filesystem root/fileset/o/*virtual device*/objects/*partition/hash_suffix/hash/object*

An example object path is:

```
/ibm/gpfs/objfs/o/z1device111/objects/11247/73a/afbeca778982b05b9dddf4fed88f773a/
1461036399.66296.data
```

Similarly, account and container databases are placed in the following directory paths:

gpfs filesystem root/fileset/ac/*virtual device*/containers/*partition/hash_suffix/hash/account db*
and

gpfs filesystem root/fileset/ac/*virtual device*/containers/*partition/hash_suffix/hash/container db*.

An example account path is:

```
/ibm/gpfs/objfs/ac/z1device62/accounts/13700/f60/d61003e46b4945e0bbbfcee341d30f60/
d61003e46b4945e0bbbfcee341d30f60.db
```

An example container path is:

```
/ibm/gpfs/objfs/ac/z1device23/containers/3386/0a9/34ea8d244872a1105b7df2a2e6ede0a9/
34ea8d244872a1105b7df2a2e6ede0a9.db
```

Starting at the bottom of the object path and working upward, each new object that is created requires a new hash directory and a new object file, thereby consuming two inodes. Similarly, for account and container data, each new account and container require a new hash directory and a db file. Also, a db.pending and a lock file is required to serialize access. Therefore, four inodes are consumed for each account and each container at the hash directory level.

If the parent directories do not already exist, they are created, thereby consuming additional inodes. The hash suffix directory is three hexadecimal characters, so there can be a maximum of 0xFFF or 4096 suffix directories per partition. The total number of partitions is specified during initial configuration. For IBM Spectrum Scale, 16384 partitions are allocated to objects and the same number is allocated to accounts and containers.

For each object partition directory, the hashes.pkl file is created to track the contents of the partition subdirectories. Also, there is a .lock file that is created for each partition directory to serialize updates to hashes.pkl. This is a total of three inodes required for each object partition.

There are 128 virtual devices allocated to object data during initial configuration, and the same number is allocated to account and container data. For each virtual device a tmp directory is created to store objects during upload. In the async_pending directory, container update requests that time out are stored until they are processed asynchronously by the object updater service.

The total number of inodes used for object storage in the traditional object layout can be estimated as follows:
```
total required inodes = account & container inodes + object inodes
```

As per this information, there are four inodes per account hash directory and four inodes per container hash directory. In the worst case, there would be one suffix directory, one partition directory, and one virtual device directory for each account and container. Therefore, the maximum inodes for accounts and containers can be estimated as:

```
account and container inodes = (7 * maximum number of accounts) + (7 * maximum number of containers)
```

In a typical object store there are more objects than containers, and more containers than accounts. Therefore, while estimating the required inodes, we estimate the number of inodes required for accounts and containers to be seven times the maximum number of containers. The maximum required inodes can be calculated as shown below:

```
max required inodes = (inodes for objects and hash directory) + (inodes required for hash directories) +
        (inodes required for partition directories and partition metadata) +
        (inodes required for virtual devices) + (inodes required for containers)
max required inodes = (2 x maximum number of objects) + (4096 inodes per partition * 16384 partitions) +
        (16384 partitions * 3) + (128 inodes) + (7 * maximum number of containers)
```

Important: As the number of objects grows, the inode requirement is dominated by the number of objects. A safe rule of thumb is to allocate three inodes per expected object when there are 10M to 100M expected objects. For more than 100M, you can allocate closer to 2.5 inodes per object.

Note: This applies to a case when all objects as well as account and container data are in the same fileset. While using multiple storage policy filesets or a different fileset for account and container data, the calculations must be adjusted.

Migration of CNFS clusters to CES clusters

If your system has established clustered Network File System (CNFS) clusters, you might consider migrating these clusters to Cluster Export Services (CES) clusters.

Points to consider before you migrate

Cluster Export Services (CES) protocol nodes have the following dependencies and restrictions:
- CES nodes cannot coexist with CNFS clusters.
- The concepts of failover in CES node groups and CNFS failover groups are slightly different. While CNFS allows to failover not just within a group but also within ranges, CES does not. Make sure that your failover concepts are handled correctly by CES.
- The installation toolkit confirms that your cluster does not have an existing CNFS cluster.
- CES nodes use SMB, NFS, and Openstack SWIFT Object services.
- File system ACL permissions need to be in NFSv4 format.
- CES NFS services expects NFSv4 ACL formats.
- CES SMB (Samba) services expects NFSv4 ACL formats.
- Existing CNFS exports definitions are not compatible with CES NFS. It is best to script and automate the creation of the equivalent exports by using the `mmnfs export add` command to reduce the amount you need to change in the future.
- CES nodes need authentication that is configured.
- There is a maximum of 16 protocol nodes in a CES cluster if the SMB protocol is also enabled.
- There is a maximum of 32 protocol nodes in a CES cluster if only NFS is enabled.

Because there is a mutual exclusivity between CNFS and CES nodes, you need to accommodate user and application access outage while CES clusters nodes are installed, configured, set up for authentication, and the NFS exports are re-created. The duration of this process depends on the complexity of the customer environment.

You might want to procure new CES nodes or reuse the existing CNFS nodes. Either way, you cannot use the installation toolkit until the CNFS nodes are unconfigured.

If you do not have an opportunity to test or plan the implementation of a CES cluster elsewhere, you might have to deal with the design and implementation considerations and issues during the planned

outage period. Usually this process is straightforward and quick. If you have a more complex environment, however, it might take longer than the allotted upgrade window to complete the migration. In this case, it is possible to set up one or two non-CNFS, NFS servers to serve NFS for a short time. During this time, you would move all your CNFS IPs to these nodes as you decommission the CNFS cluster. Then, after you successfully set up your CES nodes, authentication, and corresponding exports, you can move the IPs from the temporary NFS servers over to the CES nodes.

Saving CNFS export configuration

You need to make a copy of the exports configuration file `/etc/exports` so that you can use this file as the basis for creating the new exports in CES NFS. CES NFS exports configuration needs to be created by using the **mmnfs export add** command or created in bulk by using the **mmnfs export load** command. When you unconfigure CNFS, you also need to delete the `/etc/exports` file from each of the CNFS nodes.

Steps to unconfigure CNFS

1. CES nodes can be configured only on Red Hat Enterprise Linux 7.x and SLES 12 nodes. If you are planning to convert your existing CNFS nodes to CES nodes, ensure that they are upgraded to Red Hat Enterprise Linux 7.x or SLES 12 first. It is best to upgrade the nodes first while they are running CNFS so that you can ensure that functions are the same as before you start to change over to CES nodes.
2. Ensure that you stop application and user access to the CNFS exports.
3. Run the **mmchnode** command to dereference or evict a CNFS node from the cluster. This command removes both the node and its associated IP):

   ```
   mmchnode --cnfs-interface=default -N node1Name,node2Name,...
   ```
4. When you remove the last node, CNFS is unconfigured and you see an output similar to this result:

   ```
   [root@esnode3 ~]# mmlscluster --cnfs

   GPFS cluster information
   ========================
     GPFS cluster name:         esvcluster1.esnode1
     GPFS cluster id:           156354457952275488305

   mmlscluster:  CNFS is not defined in this cluster.

   [root@esnode3 ~]#
   ```
5. Consider de-refencing the GPFS variable *cnfsSharedRoot*, although this step is not a requirement.
6. You can now delete the `/etc/exports` file on each of the CNFS nodes. Ensure that you have a backup copy of this file to use as a reference when you create the exports under CES NFS.

Steps to Configure CES NFS

1. If you have not yet configured the CES nodes for authentication, complete this step before you create the exports. Refer to "CES NFS support" on page 463 for details on configuring authentication for your environment.
2. Ensure that the file systems you want to export access to are configured for the NFSv4 security model. If you are converting an existing file system from another security model to NFSv4 you might need to review the ACL structures of the files and verify that your access will work as expected.
3. If you have special configurations or options set in CNFS server, you might also want to reflect these settings in CES NFS. You need to review the appropriateness of these settings for the new environment. To change the settings, use the following command:

   ```
   mmnfs config change
   ```
4. If you have many exports to be converted to CES NFS, use the following command:

   ```
   mmnfs export load ExportCfgFile
   ```

ExportCfgFile contains a listing of all your exports as defined in the format that is used for
`/etc/ganesha/gpfs.ganesha.exports.conf`.

5. Alternately, you can manually re-create each export on the CES cluster by using the **mmnfs** command.

   `mmnfs export add `*`Path`*` --client `*`ClientOptions`*

6. Before you proceed to configure CES nodes, remove the NFS exports from `/etc/exports` from each of the old CNFS nodes

7. Add the IPs that were previously assigned to CNFS to the address pool to be managed by CNFS by using the following command:

   `mmces address add --node node1Name --ces-ip `*`ipAddress`*

   See "CES network configuration" on page 460 for details about how to use this command.

8. Take care to ensure that the IP addresses are unique and valid for your subnet.

For more information on creating protocol data exports, see *Fileset considerations for creating protocol data exports* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Test access to new exports on CES NFS

Test and verify that you have the same level of access to the NFS exports as you did on CNFS to ensure that your applications and NFS clients can continue without further changes.

Chapter 32. Identity management on Windows

GPFS allows file sharing among AIX, Linux, and Windows nodes. AIX and Linux rely on 32-bit user and group IDs for file ownership and access control purposes, while Windows uses variable-length security identifiers (SIDs). The difference in the user identity description models presents a challenge to any subsystem that allows for heterogeneous file sharing.

GPFS uses 32-bit ID namespace as the canonical namespace, and Windows SIDs are mapped into this namespace as needed. Two different mapping algorithms are used (depending on system configuration):

* GPFS built-in auto-generated mapping
* User-defined mappings stored in the Microsoft Windows Active Directory using the Microsoft Identity Management for UNIX (IMU) component

Auto-generated ID mappings

Auto-generated ID mappings are the default. If no explicit mappings are created by the system administrator in the Active Directory using Microsoft Identity Management for UNIX (IMU), all mappings between security identifiers (SIDs) and UNIX IDs will be created automatically using a reserved range in UNIX ID space.

Note: If you have a mix of GPFS running on Windows and other Windows clients accessing the integrated SMB server function, the ability to share data between these clients has not been tested or validated. With protocol support, the SMB server may also be configured to automatically generate ID mapping. If you want to ensure that SMB users do not access data (share ID mapping) with Windows users, ensure that the automatic range for SMB server is different from this range. The range of IDs automatically generated for the SMB server can be controlled by **mmuserauth**.

Unless the default reserved ID range overlaps with an ID already in use, no further configuration is needed to use the auto-generated mapping function. If you have a specific file system or subtree that are only accessed by user applications from Windows nodes (even if AIX or Linux nodes are used as NSD servers), auto-generated mappings will be sufficient for all application needs.

The default reserved ID range used by GPFS starts with ID 15,000,000 and covers 15,000,000 IDs. The reserved range should not overlap with any user or group ID in use on any AIX or Linux nodes. To change the starting location or the size of the reserved ID range, use the following GPFS configuration parameters:

sidAutoMapRangeLength
    Controls the length of the reserved range for Windows SID to UNIX ID mapping.

sidAutoMapRangeStart
    Specifies the start of the reserved range for Windows SID to UNIX ID mapping.

Note: For planning purposes, remember that auto-generated ID mappings are stored permanently with file system metadata. A change in the **sidAutoMapRangeStart** value is only effective for file systems created after the configuration change.

Installing Windows IMU

The Identity Management for UNIX (IMU) feature is included in Windows Server. This feature needs to be installed on the primary domain controller, as well as on any backup domain controllers. It is not installed by default. There are two components that need to be installed in order for IMU to function correctly.

When Active Directory is running on Windows Server 2008, follow these steps to add the IMU service:

1. Open Server Manager.
2. Under Roles, select **Active Directory Domain Services**.
3. Under Role Services, select **Add Role Services**.
4. Under the Identity Management for UNIX role service, check **Server for Network Information Services**.
5. Click **Next**, then **Install**.
6. Restart the system when the installation completes.

Configuring ID mappings in IMU

To configure ID mappings in Microsoft Identity Management for UNIX (IMU), follow the steps in this procedure.

1. Open **Active Directory Users and Computers** (accessible under Administrative Tools).
2. Select the **Users** branch in the tree on the left under the branch for your domain to see the list of users and groups in this domain.
3. Double-click on any user or group line to bring up the Properties window. If IMU is set up correctly, there will be a **UNIX Attributes** tab as shown in Figure 16:
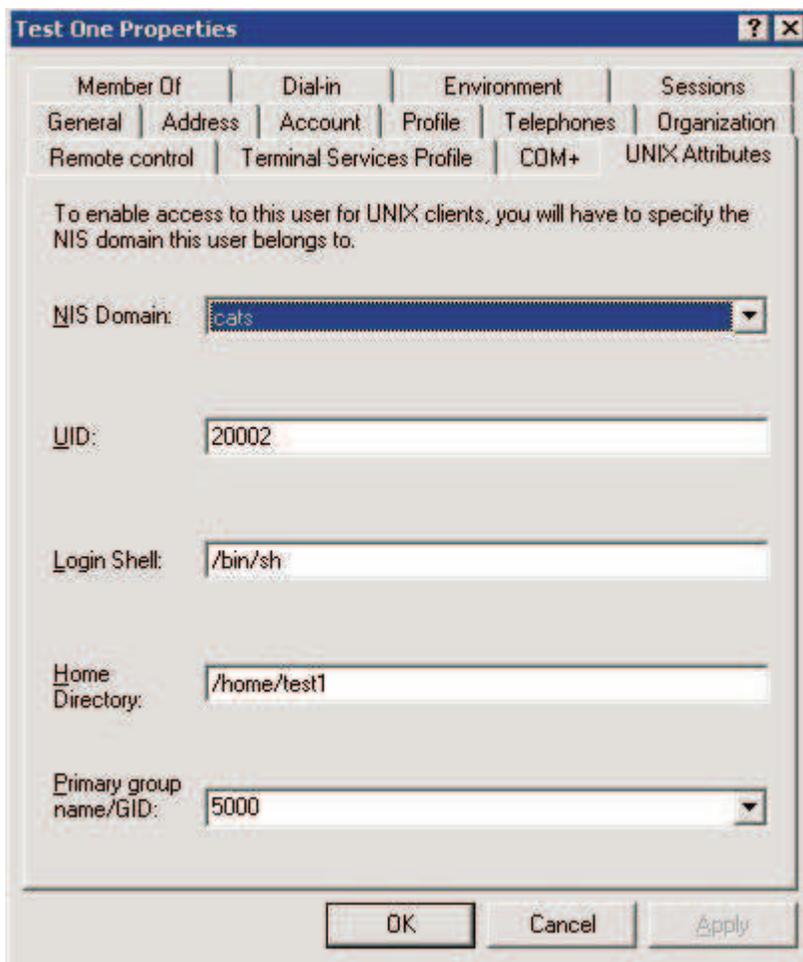
Figure 16. Properties window

Note: Because the IMU subsystem was originally designed to support integration with the UNIX Network Information Service (NIS), there is an **NIS Domain** field in the Properties window. You do not need to have NIS set up on the UNIX side. For GPFS, the NIS language does not matter.

Update information on the **UNIX Attributes** panel as follows:

1. Under the **NIS Domain** drop-down list, select the name of your Active Directory domain. Selecting **<none>** will remove an existing mapping.
2. Specify a UID in the **UID** field, and for Group objects, specify a GID. This will create a bidirectional mapping between the corresponding SID and a UNIX ID. IMU will disallow the use of the same UID or GID for more than one user or group to ensure that all mappings are unique. In addition to creating mappings for domain users and groups, you can create mappings for certain built-in accounts by going to the **Builtin branch** in the Active Directory Users and Computers panel.
3. Disregard the **Primary group name/GID** field because GPFS does not use it.

It is generally better to configure all ID mappings before mounting a GPFS file system for the first time. Doing that ensures that GPFS only stores properly remapped IDs on disk. However, it is possible to add or delete mappings at any time while GPFS file systems are mounted. GPFS picks up mapping changes dynamically (the code currently checks for mapping changes every 60 seconds), and will start using them at that time.

If an IMU mapping is configured for an ID that is already recorded in some file metadata, you must proceed with caution to avoid user confusion and access disruption. Auto-generated mappings already stored in access control lists (ACLs) on disk will continue to map correctly to Windows SIDs. However, because the SID is now mapped to a different UNIX ID, when you access a file with an ACL containing its auto-generated ID, this access will effectively appear to GPFS as an access by a different user. Depending on the file access permissions, you might not be able to access files that were previously accessible. Rewriting affected ACLs after setting up a new mapping will help replace auto-generated IDs with IMU-mapped IDs, and will restore proper file access for the affected ID (this operation might need to be performed by the system administrator). Examining file ownership and permission information from a UNIX node (for example, using the **mmgetacl** command) is the easiest way to determine whether the ACL for a specified file contains auto-generated or IMU-mapped IDs.

Chapter 33. Protocols cluster disaster recovery

Protocols cluster disaster recovery (DR) uses the capabilities of Active File Management (AFM) based Async Disaster Recovery (AFM DR) features to provide a solution that allows an IBM Spectrum Scale cluster to fail over to another cluster and fail back, and backup and restore the protocol configuration information in cases where a secondary cluster is not available.

For more information on AFM-based Async DR, see the topic *AFM-based Asynchronous Disaster Recovery (AFM DR)* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Important: Our initial feedback from the field suggests that success of a disaster recovery solution depends on administration discipline, including careful design, configuration and testing. Considering this, IBM has decided to disable the Active File Management-based Asynchronous Disaster Recovery feature (AFM DR) by default and require that customers deploying the AFM DR feature first review their deployments with IBM Spectrum Scale development. You should contact IBM Spectrum Scale Support at scale@us.ibm.com to have your use case reviewed. IBM will help optimize your tuning parameters and enable the feature. Please include this message while contacting IBM Support.

These limitations do not apply to base AFM support. These apply only to Async DR available with the IBM Spectrum Scale Advanced Edition V4.2 and V4.1.1.

For more information, see Flash (Alert): IBM Spectrum Scale (GPFS) V4.2 and V4.1.1 AFM Async DR requirement for planning.

Although an overview of the steps that need to be done is provided if performing these operations manually, it is recommended to use the **mmcesdr** command because it automates DR setup, failover, failback, backup, and restore actions. For more information about the **mmcesdr** command, see *mmcesdr command* in *IBM Spectrum Scale: Command and Programming Reference* .

Protocols cluster disaster recovery limitations and prerequisites

For protocols cluster disaster recovery (DR) in an IBM Spectrum Scale cluster, the prerequisites and limitations are as follows.

Ensure that the following prerequisites are met for the secondary cluster for disaster recovery in an IBM Spectrum Scale with protocols.
- IBM Spectrum Scale is installed and configured.
- IBM Spectrum Scale code levels are the same on the primary and secondary clusters.
- IBM Spectrum Scale code levels are the same on every protocol node within a cluster.
- Cluster Export Services (CES) are installed and configured, and the shared root file system is defined.
- All protocols that are configured on the primary cluster are also configured on the secondary cluster.
- Authentication on secondary cluster is identical to the authentication on the primary cluster.
- All exports that need to be protected using AFM DR must have the same device and fileset name, and the same fileset link point on the secondary cluster as defined on the primary cluster.
- IBM NFSv3 stack must be configured on home cluster for the AFM DR transport of data.
- No data must be written to exports on secondary cluster while cluster is acting only as a secondary cluster, before a failover.

The following limitations apply for disaster recovery in an IBM Spectrum Scale cluster with protocols.

- Only data contained within independent filesets can be configured for AFM based Async Disaster Recovery (AFM DR). Therefore, all protocol exports that you want to be protected by DR must have the export path equal to the independent fileset link point.
- Nested independent or dependent filesets are not supported.
- Backup and restore of the authentication configuration is not supported.
- On failover and failback or restore, all clients need to disconnect and then reconnect.
- If `--file-config --restore` is specified, perform the follow steps:
  - On failover: file authentication must be removed and then reconfigured on the secondary cluster.
  - On restore: file authentication must be removed and then reconfigured on the primary cluster.
  - On failback: file authentication must be removed and then reconfigured on both primary and secondary clusters.
- Multi-region object deployment is not supported with protocols cluster DR. Therefore, if multi-region object deployment is enabled, object data or configuration information is not protected through protocols cluster DR.
- IBM Spectrum Protect for Space Management and IBM Spectrum Archive migrated data within protocol exports is not supported within protocols cluster DR.
- IBM Spectrum Protect configuration file information is not automatically protected through protocols cluster DR.

Example setup for protocols disaster recovery

The following example scenario is used to show how to set up disaster recovery functionality for an IBM Spectrum Scale cluster with protocols.

This example consists of three NFS exports, three SMB shares, one object fileset, and two unified file and object access filesets that are also NFS exports. For the SMB and NFS exports, only two of each are independent filesets. This allows an AFM-based Async DR (AFM DR) configuration. For simplification, the filesets are named according to whether or not they were dependent or independent for the SMB and NFS exports. The inclusion of dependent filesets as exports is to show the warnings that are given when an export path is not an independent fileset link point.

Note: SMB and NFS exports must be named according to their fileset link point names for them to be captured by the **mmcesdr** command for protocols cluster disaster recovery. For example, if you have a fileset nfs-smb-combo, the NFS or the SMB export name must be *GPFS_Path*/nfs-smb-combo. If you use a name in the fileset's subdirectory for the SMB or the NFS export (for example: *GPFS_Path*/nfs-smb-combo/nfs1), the **mmcesdr** command does not capture that export.

NFS exports
- /gpfs/fs0/nfs-ganesha-dep
- /gpfs/fs0/nfs-ganesha1
- /gpfs/fs0/nfs-ganesha2

SMB shares
- /gpfs/fs0/smb1
- /gpfs/fs0/smb2
- /gpfs/fs0/smb-dep

Combination SMB and NFS exports
- /gpfs/fs0/combo1
- /gpfs/fs0/combo2

Object fileset

- /gpfs/fs1/object_fileset

Unified file and object access filesets

- /gpfs/fs1/obj_sofpolicy1

  This fileset is created by creating a storage policy using the **mmobj policy create sofpolicy1 --enable-file-access** command.

- /gpfs/fs1/obj_sofpolicy2

  This fileset is created by creating a storage policy using the **mmobj policy create sofpolicy2 --enable-file-access --enable-compression --compression-schedule "30:02:*:*"** command.

Setting up gateway nodes to ensure cluster communication during failover

Both the primary and the DR clusters require designating gateway nodes for access to whichever side is acting as the cache. By designating gateway nodes on both clusters, you can ensure that even during failover, cluster communication continues properly.

To handle a possible node failure, you need to specify at least two nodes on each cluster to be gateway nodes. To specify two nodes on the primary cluster as gateway nodes, use the command similar to the following:

mmchnode -N Node1,Node2 --gateway

Using the example setup mentioned in "Example setup for protocols disaster recovery" on page 478, the command to specify gateway nodes on the primary cluster is as follows:

```
# mmchnode -N zippleback-vm1,zippleback-vm2 --gateway
Tue Apr 28 20:59:01 MST 2015: mmchnode: Processing node zippleback-vm2
Tue Apr 28 20:59:01 MST 2015: mmchnode: Processing node zippleback-vm1
mmchnode: Propagating the cluster configuration data to all
affected nodes.  This is an asynchronous process.
# Tue Apr 28 20:59:04 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation started

Tue Apr 28 20:59:08 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation completed; mmdsh rc=0
```

Similarly, you need to specify at least two nodes on the DR cluster as gateway nodes. Using the example setup, the command to specify gateway nodes on the DR cluster is as follows:

```
# mmchnode -N windwalker-vm1,windwalker-vm2 --gateway
Tue Apr 28 20:59:49 MST 2015: mmchnode: Processing node windwalker-vm2
Tue Apr 28 20:59:49 MST 2015: mmchnode: Processing node windwalker-vm1
mmchnode: Propagating the cluster configuration data to all
affected nodes.  This is an asynchronous process.
# Tue Apr 28 20:59:51 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation started

Tue Apr 28 20:59:54 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation completed; mmdsh rc=0
```

Creating the inband disaster recovery setup

Use the following steps for inband disaster recovery setup in an IBM Spectrum Scale cluster with protocols.

1. On the primary cluster, use the following command to configure independent fileset exports as AFM DR filesets and back up configuration information.

```
mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 15 --inband
```

The system displays output similar to the following:

```
Performing step 1/5, configuration fileset creation/verification.
Successfully completed step 1/5, configuration fileset creation/verification.
Performing step 2/5, protocol and export services configuration backup.
Successfully completed step 2/5, protocol and export services configuration backup.
Performing step 3/5, determination of protocol exports to protect with AFM DR.
WARNING: Export /gpfs/fs0/nfs-ganesha-dep of type nfs will NOT be protected
through AFM DR because it is a dependent fileset.
Not all exports of type NFS-ganesha will be protected through AFM DR, rc: 2
WARNING: Export /gpfs/fs0/smb-dep of type smb will NOT be protected
through AFM DR because it is a dependent fileset.
Not all exports of type SMB will be protected through AFM DR, rc: 2
Completed with errors step 3/5, determination of protocol exports to protect with AFM DR.
Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
Performing step 5/5, creation of output DR configuration file.
Successfully completed step 5/5, creation of output DR configuration file.

File to be used with secondary cluster in next step of cluster DR setup: /root//DR_Config
```

Note: In this command example, there are two exports that are not protected. During the
configuration step, any exports that are not protected through AFM DR generate a warning to the
standard output of the command.

2. Use the following command to transfer the DR configuration file from the primary cluster to the
secondary cluster.

```
scp /root//DR_Config windwalker-vm1:/root/
```

The system displays output similar to the following:

```
root@windwalker-vm1's password:
DR_Config 100% 1551 1.5KB/s 00:00
```

3. On the secondary cluster, use the following command to create the independent filesets that is a part
of the pair of AFM DR filesets associated with those on the primary cluster. In addition to creating
filesets, this command also creates the necessary NFS exports.

```
mmcesdr secondary config --input-file-path /root/ --inband
```

The system displays output similar to the following:

```
Performing step 1/3, creation of independent filesets to be used for AFM DR.
Successfully completed step 1/3, creation of independent filesets to be used for AFM DR.
Performing step 2/3, creation of NFS exports to be used for AFM DR.
Successfully completed step 2/3, creation of NFS exports to be used for AFM DR.
Performing step 3/3, conversion of independent filesets to AFM DR secondary filesets.
Successfully completed step 3/3, conversion of independent filesets to AFM DR secondary filesets.
```

4. Ensure that all of the expected AFM DR pairs show as `Active` in the output of the **mmafmctl** command
and take corrective action if they do not.

```
# mmafmctl fs0 getstate

Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
------------ -------------- ------------- ------------ ------------ -------------
nfs-ganesha1 nfs://9.11.102.210/gpfs/fs0/nfs-ganesha1 Active zippleback-vm2 0 4
nfs-ganesha2 nfs://9.11.102.211/gpfs/fs0/nfs-ganesha2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4
combo1 nfs://9.11.102.210/gpfs/fs0/combo1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 7
combo2 nfs://9.11.102.211/gpfs/fs0/combo2 Active zippleback-vm2 0 66
smb1 nfs://9.11.102.210/gpfs/fs0/smb1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 65
smb2 nfs://9.11.102.211/gpfs/fs0/smb2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4

# mmafmctl fs1 getstate

Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
------------ -------------- ------------- ------------ ------------ -------------
object_fileset nfs://9.11.102.211/gpfs/fs1/object_fileset Active zippleback-vm1.tuc.stglabs.ibm.com 0 95671
obj_sofpolicy1 nfs://9.11.102.211/gpfs/fs1/obj_sofpolicy1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 27
obj_sofpolicy2 nfs://9.11.102.210/gpfs/fs1/obj_sofpolicy2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 26
async_dr nfs://9.11.102.210/gpfs/fs1/.async_dr Active zippleback-vm1.tuc.stglabs.ibm.com 0 2751
```

Note: A state of `Dirty` is normal when data is actively being transferred from the primary cluster to the secondary cluster.

Alternate Inband Set up showing the `--allowed-nfs-clients` parameter being used

With the addition of the new optional parameter **`--allowed-nfs-clients`** you can specify exactly which clients are allowed to connect to the NFS transport exports that are created on the secondary cluster. This parameter can be used for both inband and outband set up. Here is an example of the parameter being used for an inband setup:

- On the primary cluster, run the following command to configure independent fileset exports as AFM DR filesets and to back up configuration information:

  ```
  mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo
  15 --inband --allowed-nfs-clients --gateway-nodes
  ```

This command will give the following output:

```
Performing step 1/5, configuration fileset creation/verification.
Successfully completed step 1/5, configuration fileset creation/verification.
Performing step 2/5, protocol and export services configuration backup.
Successfully completed step 2/5, protocol and export services configuration backup.
Performing step 3/5, determination of protocol exports to protect with AFM DR.
WARNING: Export /gpfs/fs0/nfs-ganesha-dep of type nfs will NOT be protected through AFM DR because
     it is a dependent fileset.
Not all exports of type NFS-ganesha will be protected through AFM DR, rc: 2
WARNING: Export /gpfs/fs0/smb-dep of type smb will NOT be protected through AFM DR because
     it is a dependent fileset.
Not all exports of type SMB will be protected through AFM DR, rc: 2
Completed with errors step 3/5, determination of protocol exports to protect with AFM DR.
Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
Performing step 5/5, creation of output DR configuration file.
Successfully completed step 5/5, creation of output DR configuration file.
```

Creating the outband disaster recovery setup

Use the following steps for outband disaster recovery setup in an IBM Spectrum Scale cluster with protocols.

1. On the primary cluster, use the following command to configure independent fileset exports as AFM DR filesets and back up configuration information.

   ```
   mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 15
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/5, configuration fileset creation/verification.
   Successfully completed step 1/5, configuration fileset creation/verification.
   Performing step 2/5, protocol and export services configuration backup.
   Successfully completed step 2/5, protocol and export services configuration backup.
   Performing step 3/5, determination of protocol exports to protect with AFM DR.
   Successfully completed step 3/5, determination of protocol exports to protect with AFM DR.
   Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
   Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
   Performing step 5/5, creation of output DR configuration file.
   Successfully completed step 5/5, creation of output DR configuration file.

   File to be used with secondary cluster in next step of cluster DR setup: /root//DR_Config
   ```

2. Use the following command to transfer the DR configuration file from the primary cluster to the secondary cluster.

   ```
   scp /root//DR_Config windwalker-vm1:/root/
   ```

   The system displays output similar to the following:

   ```
   root@windwalker-vm1's password:
   DR_Config 100% 2566 2.5KB/s 00:00
   ```

3. On the secondary cluster, use the following command to create the independent filesets that will later be paired with those on the primary cluster to form AFM DR pairs.

```
mmcesdr secondary config --input-file-path /root --prep-outband-transfer
```

The system displays output similar to the following:

```
Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
Transfer all data on primary cluster for fileset fs0:combo1 to fileset fs0:combo1 on secondary
cluster.
Transfer all data on primary cluster for fileset fs0:combo2 to fileset fs0:combo2 on secondary
cluster.
Transfer all data on primary cluster for fileset fs0:nfs-ganesha1 to fileset fs0:nfs-ganesha1 on
secondary cluster.
Transfer all data on primary cluster for fileset fs0:nfs-ganesha2 to fileset fs0:nfs-ganesha2 on
secondary cluster.
Transfer all data on primary cluster for fileset fs0:smb1 to fileset fs0:smb1 on secondary cluster.
Transfer all data on primary cluster for fileset fs0:smb2 to fileset fs0:smb2 on secondary cluster.
Transfer all data on primary cluster for fileset fs1:async_dr to fileset fs1:async_dr on secondary
cluster.
Transfer all data on primary cluster for fileset fs1:obj_sofpolicy1 to fileset fs1:obj_sofpolicy1 on
secondary cluster.

Transfer all data on primary cluster for fileset fs1:obj_sofpolicy2 to fileset fs1:obj_sofpolicy2 on
secondary cluster.

Transfer all data on primary cluster for fileset fs1:object_fileset to fileset fs1:object_fileset on
secondary cluster.

Successfully completed creating independent filesets to be used as recipients of AFM DR outband
transfer of data.
Transfer data from primary cluster through outbound trucking to the newly created independent filesets
before proceeding to the next step.
```

4. Transfer the data from all protected filesets on the primary cluster to the corresponding filesets on the secondary cluster. If transferring files for which GPFS extended attributes also need to be transferred (such as object data), you must use a method that transfers GPFS extended attributes. For this purpose, you can use IBM Spectrum Protect (to back up data to tape and then restore it), GPFS cross-cluster mount, and AFM. Standard SCP and standard rsync do not transfer GPFS extended attributes.

5. After all the data has been transferred to the secondary cluster, use the following command to complete the setup on the secondary cluster.

```
mmcesdr secondary config --input-file-path /root
```

The system displays output similar to the following:

```
Performing step 1/3, verification of independent filesets to be used for AFM DR.
Successfully completed 1/3, verification of independent filesets to be used for AFM DR.
Performing step 2/3, creation of NFS exports to be used for AFM DR.
Successfully completed step 2/3, creation of NFS exports to be used for AFM DR.
Performing step 3/3, conversion of independent filesets to AFM DR secondary filesets.
Successfully completed step 3/3, conversion of independent filesets to AFM DR secondary filesets.
```

6. Ensure that all of the expected AFM DR pairs show as Active in the output of the **mmafmctl** command and take corrective action if they do not.

```
# mmafmctl fs0 getstate

Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
------------ -------------- ------------- ------------ ------------ -------------
nfs-ganesha1 nfs://9.11.102.210/gpfs/fs0/nfs-ganesha1 Active zippleback-vm2 0 4
nfs-ganesha2 nfs://9.11.102.211/gpfs/fs0/nfs-ganesha2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4
combo1 nfs://9.11.102.210/gpfs/fs0/combo1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 7
combo2 nfs://9.11.102.211/gpfs/fs0/combo2 Active zippleback-vm2 0 66
smb1 nfs://9.11.102.210/gpfs/fs0/smb1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 65
smb2 nfs://9.11.102.211/gpfs/fs0/smb2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4

# mmafmctl fs1 getstate

Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
```

```
------------ -------------- ------------- ------------ ------------ -------------
object_fileset nfs://9.11.102.211/gpfs/fs1/object_fileset Active zippleback-vm1.tuc.stglabs.ibm.com 0 95671
obj_sofpolicy1 nfs://9.11.102.211/gpfs/fs1/obj_sofpolicy1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 27
obj_sofpolicy2 nfs://9.11.102.210/gpfs/fs1/obj_sofpolicy2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 26
async_dr nfs://9.11.102.210/gpfs/fs1/.async_dr Active zippleback-vm1.tuc.stglabs.ibm.com 0 2751
```

Note: A state of `Dirty` is normal when data is actively being transferred from the primary cluster to the secondary cluster.

Performing failover for protocols cluster when primary cluster fails

The failover procedure can use a new option to choose whether or not file protocols have their shares re-created or if the entire file protocol configuration for NFS and SMB is restored. The default is to re-create the NFS and SMB shares because it does not require removing and adding the file authentication again afterwards. The re-create option keeps existing SMB and NFS exports and customer data within their corresponding filesets, while the restore option deletes any existing SMB and NFS exports (but not data within their corresponding filesets) that were on secondary system before failover occurred. However, neither re-create nor restore affect object configuration or object data. The failover can be performed in one of the following ways:

Re-create file export configuration

The re-create option keeps existing SMB and NFS exports and customer data within their corresponding filesets. It is the default option for failover because it does not require removing and adding the file authentication afterward.

When the primary cluster fails in an IBM Spectrum Scale cluster with protocols, you can fail over to the secondary cluster and re-create the file export configuration.

Do the following steps:

On the secondary cluster, after the primary cluster fails, issue the following command.

```
mmcesdr secondary failover
```

The system displays output similar to the following example:

```
Performing step 1/4, saving current NFS configuration to restore after failback.
Successfully completed step 1/4, saving current NFS configuration to restore after failback.
Performing step 2/4, failover of secondary filesets to primary filesets.
Successfully completed step 2/4, failover of secondary filesets to primary filesets.
Performing step 3/4, protocol configuration restore.
Successfully completed step 3/4, protocol configuration restore.
Performing step 4/4, create/verify NFS AFM DR transport exports.
Successfully completed step 4/4, create/verify NFS AFM DR transport exports.
```

Restore file export configuration

The restore option restores the entire file protocol configuration for NFS and SMB. It deletes any existing SMB and NFS exports that were on the secondary system before failover occurred. It does not delete the data within the corresponding filesets.

When the primary cluster fails in an IBM Spectrum Scale cluster with protocols, you can fail over to the secondary cluster and restore the file export configuration.

Do the following steps:

1. On the secondary cluster, after the primary cluster fails, issue the following command.

   ```
   mmcesdr secondary failover --file-config --restore
   ```

   The system displays output similar to the following example:

```
Performing step 1/4, saving current NFS configuration to restore after failback.
Successfully completed step 1/4, saving current NFS configuration to restore after failback.
Performing step 2/4, failover of secondary filesets to primary filesets.
Successfully completed step 2/4, failover of secondary filesets to primary filesets.
Performing step 3/4, protocol configuration/exports restore.
Successfully completed step 3/4, protocol configuration/exports restore.
Performing step 4/4, create/verify NFS AFM DR transport exports.
Successfully completed step 4/4, create/verify NFS AFM DR transport exports.

===============================================================================
= If all steps completed successfully, please remove and then re-create file
= authentication on the DR cluster.
= Once this is complete, Protocol Cluster Failover will be complete.
===============================================================================
```

2. Remove the file authentication on the secondary cluster after failover. Then add back the file authentication before failover is considered to be complete and client operations can resume, but point to the secondary cluster.

Performing failback to old primary for protocols cluster

When failing back to the old primary, the file protocol configuration can either be re-created or restored on the old primary. The NFS transport exports on the secondary need to be re-created or NFS configuration can be restored. The re-create option keeps existing SMB and NFS exports and customer data within their corresponding filesets, while the restore option deletes any existing SMB and NFS exports (but not data within their corresponding filesets) that were on secondary system before failover occurred. However, neither re-create nor restore affect object configuration or object data.

Re-create file protocol configuration for old primary

In the following example, file protocol configuration is re-created. To re-create the file exports on the old primary during restore, one of these commands can be run: **mmcesdr primary restore** or **mmcesdr primary restore --file-config --recreate**. The completion of failback on the secondary where the NFS transport export is re-created can also be performed by running one of these commands: **mmcesdr secondary failback --post-failback-complete** or **mmcesdr secondary failback --post-failback-complete --file-config --recreate**. Use the following steps for failing back to an old primary cluster in an IBM Spectrum Scale cluster with protocols.

1. On the old primary cluster, use the following command.

   ```
   mmcesdr primary failback --start --input-file-path "/root/"
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

   The system displays output similar to the following:

   ```
   Performing failback to primary on all AFM DR protected filesets.
   Successfully completed failback to primary on all AFM DR protected filesets
   ```

2. On the old primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 25 Mins. 20 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before stopping
   failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

3. On the secondary cluster (acting primary), quiesce all client operations.

4. On the old primary cluster, use the following command one more time.

```
mmcesdr primary failback --apply-updates --input-file-path "/root/"
```

The system displays output similar to the following:

```
Performing apply updates on all AFM DR protected filesets.
Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 0 Mins. 27 Secs.
Successfully completed failback update on all AFM DR protected filesets.
Depending on user load on the acting primary, this step may need to be performed again before stopping
failback.
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

5. On the old primary cluster, use the following command.

```
mmcesdr primary failback --stop --input-file-path "/root/"
```

The system displays output similar to the following:

```
Performing stop of failback to primary on all AFM DR protected filesets.
Successfully completed stop failback to primary on all AFM DR protected filesets.
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

6. On the old primary cluster, use the following command to restore configuration:

```
mmcesdr primary restore
```

The system displays output similar to the following:

```
Restoring cluster and enabled protocol configurations/exports.
Successfully completed restoring cluster and enabled protocol configurations/exports.
```

7. On the secondary cluster (acting primary), use the following command to convert it back to a secondary cluster and associate it with the original primary cluster:

```
mmcesdr secondary failback --post-failback-complete
```

The system displays output similar to the following:

```
Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
Successfully completed step 1/2, converting protected filesets back into AFM DR secondary
filesets.
Performing step 2/2, restoring/recreating AFM DR-based NFS share configuration.
Successfully completed step 2/2, restoring/recreating AFM DR-based NFS share configuration.

================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Secondary cluster.
= Once this is complete, Protocol Cluster Failback will be complete.
================================================================================
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

Restore file protocol configuration for old primary

In the following example, file protocol configuration is restored. To restore the file exports on the old primary during restore, run the following command: **mmcesdr primary restore --file-config --restore**. The completion of failback on the secondary where the NFS transport export is re-created can also be performed by running this commands: **mmcesdr secondary failback --post-failback-complete --file-config --restore**. Use the following steps for failing back to an old primary cluster in an IBM Spectrum Scale cluster with protocols.

1. On the old primary cluster, use the following command.

```
mmcesdr primary failback --start --input-file-path "/root/"
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

The system displays output similar to the following:

```
Performing failback to primary on all AFM DR protected filesets.
Successfully completed failback to primary on all AFM DR protected filesets
```

2. On the old primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

```
mmcesdr primary failback --apply-updates --input-file-path "/root/"
```

The system displays output similar to the following:

```
Performing apply updates on all AFM DR protected filesets.
Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 25 Mins. 20 Secs.
Successfully completed failback update on all AFM DR protected filesets.
Depending on user load on the acting primary, this step may need to be performed again before stopping
failback.
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

3. On the secondary cluster (acting primary), quiesce all client operations.

4. On the old primary cluster, use the following command one more time.

```
mmcesdr primary failback --apply-updates --input-file-path "/root/"
```

The system displays output similar to the following:

```
Performing apply updates on all AFM DR protected filesets.
Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 0 Mins. 27 Secs.
Successfully completed failback update on all AFM DR protected filesets.
Depending on user load on the acting primary, this step may need to be performed again before stopping
failback.
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

5. On the old primary cluster, use the following command.

```
mmcesdr primary failback --stop --input-file-path "/root/"
```

The system displays output similar to the following:

```
Performing stop of failback to primary on all AFM DR protected filesets.
Successfully completed stop failback to primary on all AFM DR protected filesets.
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

6. On the old primary cluster, use the following command to restore configuration:

```
mmcesdr primary restore --file-config --restore
```

The system displays output similar to the following:

```
Restoring cluster and enabled protocol configurations/exports.
Successfully completed restoring cluster and enabled protocol configurations/exports.

================================================================================
=  If all steps completed successfully, remove and then re-create file
=  authentication on the Primary cluster.
=  Once this is complete, Protocol Cluster Configuration Restore will be complete.
================================================================================
```

7. On the primary cluster, remove the file authentication and then add it again.

8. On the secondary cluster (acting primary), use the following command to convert it back to a secondary cluster and associate it with the original primary cluster.

```
mmcesdr secondary failback --post-failback-complete --input-file-path /root --file-config --restore
```

The system displays output similar to the following:

```
Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
Successfully completed step 1/2, converting protected filesets back into AFM DR secondary
filesets.
```

```
Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
Performing step 2/2, restoring/recreating AFM DR-based NFS share configuration.
Successfully completed step 2/2, restoring/recreating AFM DR-based NFS share configuration.


================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Secondary cluster.
= Once this is complete, Protocol Cluster Failback will be complete.
================================================================================
```

> **Note:** The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

9. On the secondary cluster, remove the file authentication and then add it again.

Performing failback to new primary for protocols cluster

When failing back to the new primary, file protocol configuration can either be re-created or restored on the new primary. The NFS transport exports on the secondary need to be re-created or NFS configuration can be restored. The re-create option keeps existing SMB and NFS exports and customer data within their corresponding filesets, while the restore option deletes any existing SMB and NFS exports (but not data within their corresponding filesets) that were on secondary system before failover occurred. However, neither re-create nor restore affect object configuration or object data.

Re-create file protocol configuration for new primary

The file protocol configuration is re-created in the following example. To re-create the file exports on the new primary during restore, one of these commands can be run:
mmcesdr primary restore or **mmcesdr primary restore --file-config --recreate**.
The completion of failback on the secondary where the NFS transport exports is re-created can also be performed by running one of these commands:
mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root" or
mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root" --file-config --recreate.
Use the following steps for failing over to a new primary cluster in an IBM Spectrum Scale cluster with protocols.

1. On the old secondary cluster, use the following command to prepare recovery snapshots that contain data that will be transferred to the new primary cluster.

   ```
   mmcesdr secondary failback --generate-recovery-snapshots --output-file-path "/root/"
   --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, generating recovery snapshots for all AFM DR acting primary filesets.
   Transfer all data under snapshot located on acting primary cluster at:
   /gpfs/fs0/combo1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-1 to fileset link point of
   fileset fs0:combo1 on new primary cluster.
   Transfer all data under snapshot located on acting primary cluster at:
   /gpfs/fs0/combo2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-2 to fileset link point of
   fileset fs0:combo2 on new primary cluster.
   Transfer all data under snapshot located on acting primary cluster at:
   /gpfs/fs0/nfs-ganesha1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-3 to fileset link
   point of fileset fs0:nfs-ganesha1 on new primary cluster.
   Transfer all data under snapshot located on acting primary cluster at:
   /gpfs/fs0/nfs-ganesha2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-4 to fileset link
   point of fileset fs0:nfs-ganesha2 on new primary cluster.
   Transfer all data under snapshot located on acting primary cluster at:
   /gpfs/fs0/smb1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-5 to fileset link point of
   fileset fs0:smb1 on new primary cluster.
   Transfer all data under snapshot located on acting primary cluster at:
   /gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6 to fileset link point of
   fileset
   fs0:smb2 on new primary cluster.
   ```

Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/.async_dr/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-2 to fileset link
point of fileset fs1:async_dr on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-3 to fileset link
point of fileset fs1:obj_sofpolicy1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-4 to fileset link
point of fileset fs1:obj_sofpolicy2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/object_fileset/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-1 to fileset link
point of fileset fs1:object_fileset on new primary cluster.
Successfully completed step 1/2, generating recovery snapshots for all AFM DR acting primary
filesets.
Performing step 2/2, creation of recovery output file for failback to new primary.
Successfully completed step 2/2, creation of recovery output file for failback to new primary.

File to be used with new primary cluster in next step of failback to new primary cluster:
/root//DR_Config
```

2.  Transfer the newly created DR configuration file to the new primary cluster.

    ```
    scp /root//DR_Config zippleback-vm1:/root/
    ```

    The system displays output similar to the following:

    ```
    root@zippleback-vm1's password:
    DR_Config 100% 1996 2.0KB/s 00:00
    ```

3. On the new primary cluster, use the following command to create the independent filesets that will
    receive the data transferred from the recovery snapshots.

    ```
    mmcesdr primary failback --prep-outband-transfer --input-file-path "/root/"
    ```

    The system displays output similar to the following:

    ```
    Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
    Successfully completed creating independent filesets to be used as recipients of AFM DR outband
    transfer of data.
    ```

Transfer data from recovery snapshots through outbound trucking to the newly created independent
filesets before proceeding to the next step.

4. Transfer data from within the recovery snapshots of the secondary cluster to the new primary
    cluster.

    Note: Only one transfer is shown in the example below.

    ```
    rsync -av /gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6/*
    zippleback-vm1:/gpfs/fs0/smb2/
    ```

    The system displays output similar to the following:

    ```
    root@zippleback-vm1's password:
    sending incremental file list
    test

    sent 68 bytes received 31 bytes 15.23 bytes/sec
    total size is 0 speedup is 0.00
    ```

    Attention: When transferring files that need to also transfer GPFS extended attributes, extra steps
    are required. This example uses standard rsync which does not transfer extended attributes.

5. On the new primary cluster, use the following command to convert the independent filesets to
    primary filesets and generate a new DR configuration file that will be used on the primary cluster
    for the next steps and then transferred to the secondary cluster to be used in a later step.

    ```
    mmcesdr primary failback --convert-new --output-file-path /root/ --input-file-path /root/
    ```

    The system displays output similar to the following:

    ```
    Performing step 1/2, conversion of independent filesets into new primary filesets to be used for AFM DR.
    Successfully completed step 1/2, failback to primary on all AFM DR protected filesets.
    Performing step 2/2, creation of output file for remaining failback to new primary steps.
    ```

```
Successfully completed step 2/2, creation of output file for remaining failback to new primary steps.
```

```
File to be used with new primary cluster in next step of failback to new primary cluster: /root//DR_Config
```

6. On the new primary cluster, use the following command.

   `mmcesdr primary failback --start --input-file-path "/root/"`

   The system displays output similar to the following:

   ```
   Performing failback to primary on all AFM DR protected filesets.
   Successfully completed failback to primary on all AFM DR protected filesets.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

7. On the new primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   `mmcesdr primary failback --apply-updates --input-file-path "/root/"`

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 45 Mins. 10 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

8. On the secondary cluster (acting primary), quiesce all client operations.

9. On the new primary cluster, use the following command one more time.

   `mmcesdr primary failback --apply-updates --input-file-path "/root/"`

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 0 Mins. 16 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

10. On the new primary cluster, use the following command to stop the failback process and convert the new primary filesets to read/write.

    `mmcesdr primary failback --stop --input-file-path "/root/"`

    The system displays output similar to the following:

    ```
    Performing stop of failback to primary on all AFM DR protected filesets.
    Successfully completed stop failback to primary on all AFM DR protected filesets.
    ```

11. On the new primary cluster, use the following command to restore the protocol and export services configuration information.

    `mmcesdr primary restore --new-primary`

    Note: The --new-primary option must be used to ensure protocol configuration is restored correctly.

    The system displays output similar to the following:

    ```
    Restoring cluster and enabled protocol
            configurations/exports. Successfully completed restoring cluster and enabled protocol
            configurations/exports.
    ```

12. Transfer the updated DR configuration file from the new primary cluster to the secondary cluster.

    `scp /root//DR_Config windwalker-vm1:/root/`

    The system displays output similar to the following:

    ```
    root@windwalker-vm1's password:
    DR_Config 100% 2566 2.5KB/s 00:00
    ```

13. On the secondary cluster, use the following command to register the new primary AFM IDs to the independent filesets on the secondary cluster acting as part of the AFM DR pairs.

```
mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root"
```

The system displays output similar to the following:

```
Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
Performing step 2/2, recreating AFM DR-based NFS share configuration.
Successfully completed step 2/2, recreating AFM DR-based NFS share configuration.


================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Secondary cluster.
= Once this is complete, Protocol Cluster Failback will be complete.
================================================================================
```

Restore file protocol configuration for new primary

The file protocol configuration is restored in the following example. To restore the file exports on the old primary during restore, run the following command: **mmcesdr primary restore --file-config --restore**. The completion of failback on the secondary where the NFS transport export is re-created can also be performed by running this commands: **mmcesdr secondary failback --post-failback-complete --file-config --restore**. Use the following steps for failing back to an old primary cluster in an IBM Spectrum Scale cluster with protocols

1. On the old secondary cluster, use the following command to prepare recovery snapshots that contain data that will be transferred to the new primary cluster.

```
mmcesdr secondary failback --generate-recovery-snapshots --output-file-path "/root/"
--input-file-path "/root/"
```

The system displays output similar to the following:

```
Performing step 1/2, generating recovery snapshots for all AFM DR acting primary filesets.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/combo1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-1 to fileset link point of
fileset fs0:combo1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/combo2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-2 to fileset link point of
fileset fs0:combo2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/nfs-ganesha1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-3 to fileset link
point of fileset fs0:nfs-ganesha1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/nfs-ganesha2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-4 to fileset link
point of fileset fs0:nfs-ganesha2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-5 to fileset link point of
fileset fs0:smb1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6 to fileset link point of
fileset
fs0:smb2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/.async_dr/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-2 to fileset link
point of fileset fs1:async_dr on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-3 to fileset link
point of fileset fs1:obj_sofpolicy1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-4 to fileset link
point of fileset fs1:obj_sofpolicy2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/object_fileset/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-1 to fileset link
point of fileset fs1:object_fileset on new primary cluster.
Successfully completed step 1/2, generating recovery snapshots for all AFM DR acting primary
filesets.
```

```
Performing step 2/2, creation of recovery output file for failback to new primary.
Successfully completed step 2/2, creation of recovery output file for failback to new primary.

File to be used with new primary cluster in next step of failback to new primary cluster:
/root//DR_Config
```

2. Transfer the newly created DR configuration file to the new primary cluster.

   ```
   scp /root//DR_Config zippleback-vm1:/root/
   ```

   The system displays output similar to the following:

   ```
   root@zippleback-vm1's password:
   DR_Config 100% 1996 2.0KB/s 00:00
   ```

3. On the new primary cluster, use the following command to create the independent filesets that will receive the data transferred from the recovery snapshots.

   ```
   mmcesdr primary failback --prep-outband-transfer --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
   Successfully completed creating independent filesets to be used as recipients of AFM DR outband
   transfer of data.
   ```

Transfer data from recovery snapshots through outbound trucking to the newly created independent filesets before proceeding to the next step.

4. Transfer data from within the recovery snapshots of the secondary cluster to the new primary cluster.

   Note: Only one transfer is shown in the example below.

   ```
   rsync -av /gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6/*
   zippleback-vm1:/gpfs/fs0/smb2/
   ```

   The system displays output similar to the following:

   ```
   root@zippleback-vm1's password:
   sending incremental file list
   test

   sent 68 bytes received 31 bytes 15.23 bytes/sec
   total size is 0 speedup is 0.00
   ```

   Attention: When transferring files that need to also transfer GPFS extended attributes, extra steps are required. This example uses standard rsync which does not transfer extended attributes.

5. On the new primary cluster, use the following command to convert the independent filesets to primary filesets and generate a new DR configuration file that will be used on the primary cluster for the next steps and then transferred to the secondary cluster to be used in a later step.

   ```
   mmcesdr primary failback --convert-new --output-file-path /root/ --input-file-path /root/
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, conversion of independent filesets into new primary filesets to be used for AFM DR.
   Successfully completed step 1/2, failback to primary on all AFM DR protected filesets.
   Performing step 2/2, creation of output file for remaining failback to new primary steps.
   Successfully completed step 2/2, creation of output file for remaining failback to new primary steps.

   File to be used with new primary cluster in next step of failback to new primary cluster: /root//DR_Config
   ```

6. On the new primary cluster, use the following command.

   ```
   mmcesdr primary failback --start --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing failback to primary on all AFM DR protected filesets.
   Successfully completed failback to primary on all AFM DR protected filesets.
   ```

   Note: The `--input-file-path` parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

7. On the new primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   `mmcesdr primary failback --apply-updates --input-file-path "/root/"`

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 45 Mins. 10 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

8. On the secondary cluster (acting primary), quiesce all client operations.

9. On the new primary cluster, use the following command one more time.

   `mmcesdr primary failback --apply-updates --input-file-path "/root/"`

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 0 Mins. 16 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

10. On the new primary cluster, use the following command to stop the failback process and convert the new primary filesets to read/write.

    `mmcesdr primary failback --stop --input-file-path "/root/"`

    The system displays output similar to the following:

    ```
    Performing stop of failback to primary on all AFM DR protected filesets.
    Successfully completed stop failback to primary on all AFM DR protected filesets.
    ```

11. On the new primary cluster, use the following command to restore the protocol and export services configuration information.

    `mmcesdr primary restore --new-primary --file-config --restore`

    Note: The --new-primary option must be used to ensure protocol configuration is restored correctly.

    The system displays output similar to the following:

    ```
    Restoring cluster and enabled protocol configurations/exports.
    Successfully completed restoring cluster and enabled protocol configurations/exports.

    ===============================================================================
    =  If all steps completed successfully, remove and then re-create file
    =  authentication on the Primary cluster.
    =  Once this is complete, Protocol Cluster Configuration Restore will be complete.
    ===============================================================================
    ```

12. On the primary cluster, remove the file authentication and then add it again.

13. Transfer the updated DR configuration file from the new primary cluster to the secondary cluster.

    `scp /root//DR_Config windwalker-vm1:/root/`

    The system displays output similar to the following:

    ```
    root@windwalker-vm1's password:
    DR_Config 100% 2566 2.5KB/s 00:00
    ```

14. On the secondary cluster, use the following command to register the new primary AFM IDs to the independent filesets on the secondary cluster acting as part of the AFM DR pairs.

    ```
    mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root"
      --file-config --restore
    ```

    The system displays output similar to the following:

```
Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
Performing step 2/2, restoring AFM DR-based NFS share configuration.
Successfully completed step 2/2, restoring AFM DR-based NFS share configuration.


================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Secondary cluster.
= Once this is complete, Protocol Cluster Failback will be complete.
================================================================================
```

15. On the secondary cluster, remove the file authentication and then add it again.

Backing up and restoring protocols and CES configuration information

The backup and restore capabilities of the **mmcesdr** command can be used when there is no secondary cluster and the user still wants to protect protocol and export services configuration information. The independent fileset used to store protocol and export services configuration information can be backed up using IBM Spectrum Protect software and then restored if it ever needs to be restored to the cluster.

Note: The following steps only describe how to back up and restore the protocols and CES configuration information. The actual data contained in protocol exports would need to be backed up and restored separately.

1. On the primary cluster, use the following command to back up the configuration information:

   ```
   mmcesdr primary backup
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, configuration fileset creation/verification.
   Successfully completed step 1/2, configuration fileset creation/verification.
   Performing step 2/2, protocol and export services configuration backup.
   Successfully completed step 2/2, protocol and export services configuration backup.
   ```

   For backup, you can use IBM Spectrum Protect (formerly known as Tivoli Storage Manager) or some other tool. For example, you can use **mmbackup** as follows:

   ```
   mmbackup configuration_fileset_link_point --scope inodepace -t full
   ```

2. On the primary cluster, restore data from the off cluster storage into the configuration fileset. If **mmbackup** was used to back up the configuration fileset, the IBM Spectrum Protect command to restore is similar to the following:

   ```
   dsmc restore -subdir=yes "configuration_fileset_link_point /*"
   ```

3. On the primary cluster, use the following command to restore the configuration information:

   ```
   mmcesdr primary restore
   ```

   The system displays output similar to the following:

   ```
   Restoring cluster and enabled protocol configurations/exports.
   Successfully completed restoring cluster and enabled protocol configurations/exports.
   ```

   In some cases, running the **mmcesdr primary restore** command might display the following error message:
   `Saved configuration file does not exist..` In this case, do the following:

   - If this cluster is part of a Protocols DR relationship, place a copy of the DR configuration file at a specified location and run the **mmcesdr primary restore** command again using the `--input-file-path` option.

   - If this cluster is not part of a Protocols DR relationship, run this command again with the `--file-config --restore` option to force restoring the file configuration information. The system displays output similar to the following:

     ```
     # mmcesdr primary restore --file-config --restore
     Restoring cluster and enabled protocol configurations/exports.
     Successfully completed restoring cluster and enabled protocol configurations/exports.
     ```

```
================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Primary cluster.
= Once this is complete, Protocol Cluster Configuration Restore will be complete.
================================================================================
```

4. On the primary cluster, remove the file authentication and then add it again:

   Note: If you want to perform a restore as part of a failback (either to an old primary cluster or a new primary cluster) and want to re-create the file configuration/exports, use one of the following commands:
   mmcesdr primary restore
   or
   mmcesdr primary restore --file-config --recreate.

Updating protocols and CES configuration information

Protocol configuration information is backed up when primary configuration is run. However, if any of the protocol settings change after initial primary configuration, the saved configuration information must be updated.

You can use the following command to update the backed up configuration information for Object, NFS, and SMB protocols, and CES.

`mmcesdr primary update {--obj | --nfs | --smb | --ces}`

Note: No output is generated by the command, because this command is designed to be scripted and run on a regular basis or run as a part of a callback. The update command can only be used to update the primary configuration after Protocols DR has been configured. If no secondary cluster exists, use the **mmcesdr primary backup** command to back up configuration for later restore.

Protocols and cluster configuration data required for disaster recovery

For protocols cluster disaster recovery, data needs to be collected for failover, failback, backup, or restore from the respective protocol, for authentication, and for cluster wide information.

Use the following information to collect the data required for protocols cluster disaster recovery.

Object data required for protocols cluster DR

Data required for object protocol in case of a disaster recovery scenario is as follows.

Note: IBM Spectrum Scale 4.2 and later versions for object storage supports either AFM DR-based protection or multi-region object deployment , but not both. If multi-region object deployment is enabled, no object data or configuration information is protected through protocols cluster DR.

In addition to the standard object fileset, independent filesets are created for each object policy that is created. This in turn creates additional CCR files that are listed here. All of these additional filesets and additional configuration information are protected, if IBM Spectrum Scale for object storage is using the AFM DR-based protection and not multi-region object deployment.

You can determine all object filesets using the **mmobj policy list** command.

The object related files in CCR that need to be backed up are as follows:

1. `account.builder`
2. `account.ring.gz`
3. `account-server.conf`
4. `container.builder`

5. `container-reconciler.conf`
6. `container.ring.gz`
7. `container-server.conf`
8. `keystone_ssl.tar`
9. `keystone.tar`
10. `object.builder`
11. `object<index>.builder` for each storage policy, where *<index>* is a unique number representing the storage policy
12. `object-expirer.conf`
13. `object.ring.gz`
14. `object<index>.ring.gz` for each storage policy, where *<index>* is a unique number representing the storage policy
15. `object-server.conf`
16. `object-server-sof.conf`
17. `openrc`
18. `objRingVersion`
19. `postgresql-obj.service`
20. `proxy-server.conf`
21. `spectrum-scale-object.conf`
22. `spectrum-scale-object-policies.conf`
23. `spectrum-scale-objectizer.conf`
24. `spectrum-scale-local-region.conf`
25. `spectrum-scale-global-region.conf`. (This file is present only in multi-region deployments.)
26. `spectrum-scale-compression-scheduler.conf`
27. `spectrum-scale-compression-status.stat`
28. `swift.conf`
29. `swift.tar`

The following CCR files also need to be backed up for local object authentication :
- `keystone.conf`
- `keystone-paste.ini`
- `logging.conf`
- `wsgi-keystone.conf`

For a list of object authentication related CCR files and variables that need to be backed up, see "Authentication related data required for protocols cluster DR" on page 503.

You can back up Postgres database files as follows:
1. Take the file system snapshot of the shared root file system.
2. Create a tar or a zip file of the `object` directory within the snapshot of the CES shared root file system and everything underneath this directory.
3. Save the tar or the zip file
4. Delete the snapshot.

Failover steps for object configuration if you are using local authentication for object

Use the following steps on a protocol node in the secondary cluster to fail over the object configuration data. Use this set of steps if you are using local authentication for object.

You need to determine the following two values in the secondary cluster:

- Cluster host name: This is the DNS value which returns a CES IP address from the pool of addresses in the secondary cluster.
- Object database node: This is the CES IP address which is configured to run the postgresql-obj database for object services. You can find this value as the address designated as the object_database_node in the output of the **mmces address list** command. For example:

```
mmces address list

Address        Node      Group    Attribute
--------------------------------------------------------------
10.0.100.115   vwnode3   none     object_database_node
10.0.100.116   vwnode3   none     object_singleton_node
```

Important:

The following object steps must be run on the node designated as object_database_node in the secondary cluster. This ensures that postgresql-obj and Keystone servers can connect during this configuration process.

1. Stop the object protocol services using the following command:

   ```
   mmces service stop OBJ --all
   ```

2. Make two changes in the preserved Cluster Configuration Repository (CCR) configuration to update it for the DR environment:

   a. The keystone.conf file: Edit this file to change the database connection address to object_database_node of the secondary cluster. For example:

      ```
      Change this:
      [database]
      connection = postgresql://keystone:password@192.168.56.1/keystone

      to this:
      [database]
      connection = postgresql://keystone:password@192.168.1.3/keystone
      ```

      Note: If the **mmcesdr** command is used to save the protocol cluster configuration, then the preserved copy of the keystone.conf file is located at the following location:

      CES_shared_root_mount_point/.async_dr/Failover_Config/Object_Config/latest/ccr_files/keystone.conf

      You can edit the file directly to make this change or use the **openstack-config** command. For example, first retrieve the current value using get, and then update it using the set option:

      ```
      openstack-config --get keystone.conf  database connection \
      postgresql://keystone:passw0rd@192.168.56.1/keystone

      openstack-config --set keystone.conf  database connection \
      postgresql://keystone:passw0rd@192.168.1.3/keystone

      openstack-config --get keystone.conf  database connection \
      postgresql://keystone:passw0rd@192.168.1.3/keystone
      ```

   b. The *ks_dns_name* variable: This is one of the object related variables that was originally preserved from CCR. Modify the value of this variable to the cluster hostname of the secondary cluster.

      Note: If the **mmcesdr** command is used to save the protocol cluster configuration, then the preserved copy of the *ks_dns_name* variable is located as a line in the following file:

      CES_shared_root_mount_point/.async_dr/Failover_Config/Object_Config/latest/ccr_vars/file_for_ccr_variables.txt

      Change the value of the variable in this preserved copy of the file.

3. [Optional] If `spectrum-scale-localRegion.conf` exists from CCR, change the cluster hostname and cluster_id properties to the cluster host name and cluster id as shown in the output of the **mmlscluster** command.

4. Restore the Postgres database information to the shared root directory. The directory needs to be first cleaned out before the archive is restored. This can be done with commands similar to the following, assuming that the directory was tar/zip when it was backed up:

   a. Delete the old Postgres data:

      `rm -rf <shared_root_location>/object/keystone/*`

   b. Verify that the shared root directory is empty:

      `ls <shared_root_location>/object/keystone`

   c. Restore the current Postgres database:

      `tar xzf <tar_file_name>.gz -C <shared_root_location>`

   d. Delete the process status file from the primary:

      `rm -rf <shared_root_location>/object/keystone/postmaster.pid`

   e. List the Postgres files:

      `ls <shared_root_location>/object/keystone`

5. Restore all object configuration CCR files except `objRingVersion`, including **keystone.conf** with the modification for `object_database_node`, with a command similar to the following:

   `mmccr fput <file> <location>/<file>`

6. If object policies are present, restore all of the object policy related CCR files.

7. Restore the object configuration CCR file `objRingVersion`.

8. Restore all object configuration CCR variables, including *ks_dns_name* with the modification for cluster host name, with a command similar to the following:

   `mmccr vput name value`

9. Start the Postgres database and verify that it is running successfully using commands similar to the following:

   ```
   systemctl start postgresql-obj
   sleep 5
   systemctl status postgresql-obj
   ```

   ```
   These commands generate output similar to:
   postgresql-obj.service - postgresql-obj database server
   Loaded: loaded (/etc/systemd/system/postgresql-obj.service; disabled)
   Active: active (running) since Thu 2015-05-28 19:00:17 EDT; 20h ago
   ```

10. Load `openrc` definitions by running the following command.

    `source /root/openrc`

11. Run the following command for the value of the api_v3 pipeline. Save the output of this command into the variable <savedAPI_V3Pipeline>.
    `mmobj config list --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline` . If the value of the api_v3 pipeline does not contain the string **admin_token_auth**, then do the following:

    a. Make a note that the api_v3 pipeline had to be updated.

    b. Generate the new value of the api_v3 pipeline by inserting the string **admin_token_auth** directly after the string **token_auth** in the saved copy of the api_v3 pipeline.

    c. Change the api_v3 pipeline value using the command: `mmobj config change --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline --value <newAPI_V3Pipeline>`, where *<newAPI_V3Pipeline>* is the updated value from the previous step.

    d. List the updated api_v3 pipeline value by running the following command and ensure the **admin_token_auth** string is present: `mmobj config list --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline`.

12. Save the restored CCR copy of the keystone.conf to the local location using a command similar to **mmccr fget keystone.conf /etc/keystone/keystone.conf**. Also, update the owner and the group of this file using the following command: **chown keystone:keystone /etc/keystone/keystone.conf**.

    Note: If SSL is enabled, SSL certificates must be in place when you are saving keystone.conf from another cluster.

13. If the **DEFAULT admin_token** is set, save its current value by using a command similar to the following:

    ```
    openstack-config --get /etc/keystone/keystone.conf DEFAULT admin_token
    ```

    If a value is returned from the above command, save it because it will need to be restored later.

14. In the keystone.conf file, set **admin_token** to ADMIN using the **openstack-config** command as follows.

    ```
    openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token ADMIN
    ```

15. Set the following environment variables.

    ```
    export OS_TOKEN=ADMIN      # The value from admin_token
    export OS_URL="http://127.0.0.1:35357/v3"
    ```

16. Start Keystone services and get the list of endpoint definitions using commands similar to the following.

    ```
    systemctl start httpd
    sleep 5
    openstack endpoint list
    ```

These commands generate output similar to:

```
+------------+--------+--------------+--------------+---------+-----------+-----------------------------------------------------------------+
| ID         | Region | Service Name | Service Type | Enabled | Interface | URL                                                             |
+------------+--------+--------------+--------------+---------+-----------+-----------------------------------------------------------------+
| c36e..9da5 | None   | keystone     | identity     | True    | public    | http://specscaleswift.example.com:5000/                         |
| f4d6..b040 | None   | keystone     | identity     | True    | internal  | http://specscaleswift.example.com:35357/                        |
| d390..0bf6 | None   | keystone     | identity     | True    | admin     | http://specscaleswift.example.com:35357/                        |
| 2e63..f023 | None   | swift        | object-store | True    | public    | http://specscaleswift.example.com:8080/v1/AUTH_%(tenant_id)s    |
| cd37..9597 | None   | swift        | object-store | True    | internal  | http://specscaleswift.example.com:8080/v1/AUTH_%(tenant_id)s    |
| a349..58ef | None   | swift        | object-store | True    | admin     | http://specscaleswift.example.com:8080                          |
+------------+--------+--------------+--------------+---------+-----------+-----------------------------------------------------------------+
```

17. Update the host name specified in the endpoint definitions in the URL value from the endpoint list. The values in the endpoint table might have the cluster host name (ces1 in this example) from the primary system. They need to be updated for the cluster host name in the DR environment. In some environments, the cluster host name is the same between the primary and secondary clusters. If that is the case, skip this step.

    a. Delete the existing endpoints with the incorrect cluster host name. For each of the endpoints, use the ID value to delete the endpoint. For example, use a command similar to this to delete each of the six endpoints:

    ```
    openstack endpoint delete e149
    ```

    b. Recreate the endpoints with the cluster host name of the secondary cluster using the following commands:

    The *CHN* variable in the following commands is the cluster host name for the secondary cluster.

    ```
    openstack endpoint create identity public "http://$CHN:5000/v3"
    openstack endpoint create identity internal "http://$CHN:35357/v3"
    openstack endpoint create identity admin "http://$CHN:35357/v3"
    openstack endpoint create object-store public "http://$CHN:8080/v1/AUTH_%(tenant_id)s"
    openstack endpoint create object-store internal "http://$CHN:8080/v1/AUTH_%(tenant_id)s"
    openstack endpoint create object-store admin "http://$CHN:8080"
    ```

    c. Verify that the endpoints are now using the correct cluster host name using the following command:

    ```
    openstack endpoint list
    ```

18. If the api_v3 pipeline had to be updated previously, return it to its original value by running the following command: mmobj config change --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline --value <savedAPI_V3Pipeline>
    , where *<savedAPI_V3Pipeline>* is the value of the api_v3 pipeline saved above.

19. Depending on whether a value for the DEFAULT admin_token was previously set, do one of the following:
   - If a value for the **DEFAULT admin_token** was previously set, reset it to that value using a command similar to the following:

     ```
     openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token ${currentAdminToken}
     ```
   - If there was no previous value for the **DEFAULT admin_token**, delete the value that was set to convert the endpoint definitions using a command similar to the following:

     ```
     openstack-config --del /etc/keystone/keystone.conf DEFAULT admin_token
     ```
20. Stop the services that were manually started earlier and clean up using the following commands:

    ```
    systemctl stop httpd
    systemctl stop postgresql-obj
    unset OS_URL
    unset OS_TOKEN
    ```
21. Start the object protocol services using the following command:

    ```
    mmces service start OBJ --all
    ```

Failover steps for object configuration if you are not using local authentication for object:

Use the following steps on a protocol node on a secondary cluster to fail over object configuration data. Use this set of steps if you are not using local authentication for object.

1. Stop the object protocol services using the following command:

   ```
   mmces service stop OBJ --all
   ```
2. Restore all object configuration CCR files except `objRingVersion` with a command similar to the following:

   ```
   mmccr fput <file> <location>/<file>
   ```
3. If object policies are present, restore all of the object policy related CCR files (that is `*.builder` and `*.ring.gz` files).
4. Restore the object configuration CCR file `objRingVersion`.
5. Restore all object configuration CCR variables with a command similar to the following:

   ```
   mmccr vput name value
   ```
6. Start the object protocol services using the following command:

   ```
   mmces service start OBJ --all
   ```

Failback or restore steps for object configuration

Use the following steps on a protocol node in the primary cluster to fail back or restore the object configuration data.

You need to determine the `object_database_node` on the primary cluster once repaired or replaced.

- Object database node: This is the CES IP address which is configured to run the postgresql-obj database for object services. You can find this value as the address designated as the `object_database_node` in the output of the **mmces address list** command. For example:

  ```
  mmces address list

  Address        Node     Group   Attribute
  -------------------------------------------------------
  10.0.100.115   vwnode3  none    object_database_node
  10.0.100.116   vwnode3  none    object_singleton_node
  ```

Important:

The following object steps must be run on the node designated as `object_database_node` in the primary cluster. This ensures that postgresql-obj and Keystone servers can connect during this configuration process.

1. Stop the object protocol services using the following command:

   ```
   mmces service stop OBJ --all
   ```

2. Restore the Postgres database information to the shared root directory. The directory needs to be first cleaned out before the archive is restored. This can be done with commands similar to the following, assuming that the directory was tar/zip when it was backed up:

   a. Delete the old Postgres data:

   ```
   rm -rf <shared_root_location>/object/keystone/*
   ```

   b. Verify that the shared root directory is empty:

   ```
   ls <shared_root_location>/object/keystone
   ```

   c. Restore the current Postgres database:

   ```
   tar xzf <tar_file_name>.gz -C <shared_root_location>
   ```

   d. Delete the process status file from the primary:

   ```
   rm -rf <shared_root_location>/object/keystone/postmaster.pid
   ```

   e. List the Postgres files:

   ```
   ls <shared_root_location>/object/keystone
   ```

3. Restore all object configuration CCR files except `objRingVersion` with a command similar to the following:

   ```
   mmccr fput <file> <location>/<file>
   ```

4. If object policies are present, restore all of the object policy related CCR files (that is `*.builder` and `*.ring.gz` files).

5. Restore the object configuration CCR file `objRingVersion`.

6. Restore all object configuration CCR variables with a command similar to the following:

   ```
   mmccr vput name value
   ```

7. Start the object protocol services using the following command:

   ```
   mmces service start OBJ --all
   ```

SMB data required for protocols cluster DR

Data required for the SMB protocol in case of a disaster recovery scenario is as follows.

You can determine the SMB shares using the **mmsmb export list** command.

The SMB protocol related files that need to be backed up are as follows.
- `account_policy.tdb`
- `autorid.tdb`[1]
- `group_mapping.tdb`
- `passdb.tdb`
- `registry.tdb`
- `secrets.tdb`
- `share_info.tdb`
- `ctdb.tdb`

Note: [1] This file is required only if the file authentication is configured with Active Directory.

The following information is common for all of these files:
- The type of these files is persistent TDB.
- They are not in the cluster configuration repository (CCR).
- Their location is `/var/lib/ctdb/persistent` on all protocol nodes.
- Their contents are same on all the nodes.

- Their name consists of TDB name + node specific extension. For example: `registry.tdb.0`

The private Kerberos configuration files available at the following location also need to be backed up: `/var/lib/samba/smb_krb5/`. You can copy these files from this location and save them.

Failover steps for the SMB protocol

Use the following steps on a protocol node in the secondary cluster to fail over the SMB protocol configuration.

1. Before stopping the SMB services, make a note of the node number that will be used to restore TDB files because the node numbers are not available when SMB services are stopped.
2. Stop the SMB services using the following command:

   `mmces service stop SMB --all`
3. Issue the following command to stop the NFS service:

   `mmces service stop NFS --all`
4. Remove the contents of the `/var/lib/ctdb/persistent` directory on all protocol nodes.
5. Restore the previously saved TDB files to one of the protocol nodes and place them in the `/var/lib/ctdb/persistent` directory for the node where the node number was saved.

   However, when copying the files to that directory on the node, replace the `X.bak`, where X represents the node number where the files were copied from, with the new node number. It is crucial that each of these files ends with `.tdb.Y`, where Y is the node number that was saved and the node number where the files are being restored. These files only need to be put into one of the nodes and when the SMB processes are started again they are copied around to the other nodes properly.
6. Remove the contents of the `/var/lib/samba/smb_krb5` directory on all the protocol nodes.
7. Restore the saved contents of `smb_krb5` to the `/var/lib/samba/smb_krb5/` directory on one of the protocol nodes. No special extension needs to be altered in this case.
8. Start the SMB services using the following command:

   `mmces service start SMB --all`
9. Issue the following command to start the NFS service:

   `mmces service start NFS --all`
10. Remove the SMB shares that are not protected using AFM DR independent filesets.

Failback or restore steps for the SMB protocol

Use the following steps on a protocol node in the primary cluster, once repaired or replaced, to fail back or restore the SMB protocol configuration.

1. Stop the NFS services using the following command:

   `mmces service stop NFS --all`
2. Stop the SMB services using the following command:

   `mmces service stop smb --all`
3. Delete all files from the `/var/lib/ctdb/persistent` directory on all protocol nodes.
4. Restore all required persistent TDB files from the saved configuration location to the `/var/lib/ctdb/persistent` directory on one of the protocol nodes. Ensure that you append the node number to the end of file names.
5. Delete all private Kerberos configuration files in the `/var/lib/samba/smb_krb5/` directory on all protocol nodes.
6. Restore private Kerberos configuration files to the `/var/lib/samba/smb_krb5/` directory on one of the protocol nodes.
7. On the failback cluster, start the SMB services using the following command:

   `mmces service start smb --all`
8. Issue the following command on the failback cluster to start the NFS service:

   `mmces service start NFS --all`

NFS data required for protocols cluster DR

Data required for NFS protocol in case of a disaster recovery scenario is as follows.

To find the NFS exports, enter the following command:

```
mmnfs export list
```

The system displays output similar to the following:

```
Path            Delegations Clients
----------------------------------------
/ibm/fs1/fset1 none         10.0.0.1
/ibm/fs1/fset1 none         10.0.0.2
/ibm/fs1/fset1 none         *
```

If the NFS exports are independent filesets, AFM based Disaster Recovery (AFM DR) can be used to replicate the data.

The NFS protocol related CCR files that need to be backed up are as follows.
* `gpfs.ganesha.main.conf`
* `gpfs.ganesha.nfsd.conf`
* `gpfs.ganesha.log.conf`
* `gpfs.ganesha.exports.conf`
* `gpfs.ganesha.statdargs.conf`

The following NFS protocol related CCR variable needs to be backed up.
* *nextexportid*

Failover steps for the NFS protocol

Use the following steps on one of the protocol nodes in the secondary cluster to fail over the NFS protocol configuration.
1. Stop the NFS services using the following command:

   ```
   mmces service stop NFS --all
   ```
2. Edit the saved `gpfs.ganesha.exports.conf` export configuration file to remove all exports that are not protected through AFM DR independent filesets.
3. Restore the NFS related CCR files. For a list of these files, see "NFS data required for protocols cluster DR"
4. Restore the *nextexportid* CCR variable.
5. Load the exports file using the following command:

   ```
   mmnfs export load /<Path_to_CCR_files>/gpfs.ganesha.exports.conf
   ```
6. Start the NFS services using the following command:

   ```
   mmces service start NFS --all
   ```

Failback or restore steps for the NFS protocol

Use the following steps on a protocol node in the primary cluster, once repaired or replaced, to fail back or restore the NFS protocol configuration.
1. Stop the NFS services using the following command:

   ```
   mmces service stop NFS --all
   ```
2. Restore the NFS related CCR files. For a list of these files, see "NFS data required for protocols cluster DR."
3. Restore the *nextexportid* CCR variable.
4. Load the exports file using the following command:

   ```
   mmnfs export load /<Path_to_saved_CCR_files>/gpfs.ganesha.exports.conf
   ```

5. Start the NFS services using the following command:

```
mmces service start NFS --all
```

Authentication related data required for protocols cluster DR

Authentication data required in case of a disaster recovery scenario is as follows.

The following authentication related CCR file needs to be backed up for disaster recovery.
- `authccr`

File authentication related data

The following CCR variable needs to be backed up for file authentication:
- *FILE_AUTH_TYPE*

Depending on the file authentication scheme you are using, additional files need to be backed up.

LDAP for file authentication:
- `SSSD_CONF`
- `LDAP_CONF`
- `KRB5_CONF` [1]
- `KRB5_KEYTAB` [1]
- `LDAP_TLS_CACERT` [1]

Active Directory (AD) for file authentication:
- `KRB5_CONF`
- `KRB5_KEYTAB` [1]

NIS for file authentication:
- `SSSD_CONF`
- `YP_CONF`

Note: [1] This file is not always present.

Object authentication related data

The object authentication related files in CCR that need to be backed up are as follows:
- `keystone.conf`
- `keystone-paste.ini`
- `logging.conf`
- `wsgi-keystone.conf`
- `ks_ext_cacert.pem`
- `keystone_ssl.tar`
- `authccr`

The object authentication related variables in CCR that need to be backed up are as follows:
- *OBJECT_AUTH_TYPE*
- *PREV_OBJECT_AUTH_TYPE*
  This variable may not be present if the authentication type has not changed.
- *OBJECT_IDMAPDELETE*
- *ks_db_type*

- *ks_db_user*
- *ks_dns_name*

Failover steps for authentication data

Use the following steps on a protocol node in the secondary cluster to fail over the authentication configuration.

Note: The object authentication does not need to be removed and re-added as part of failover, failback, or restore.

1. Save the current file authentication information on the secondary cluster.
2. Remove file authentication from the secondary cluster.
3. Restore file authentication on the secondary cluster based on the information saved in step 1.

Failback steps for authentication data

Use the following steps on a protocol node in the primary cluster, once repaired or replaced, to fail back the authentication configuration.

Note: The object authentication does not need to be removed and re-added as part of failover, failback, or restore.

1. Save the current file authentication information on the primary cluster.
2. Remove file authentication from the primary cluster.
3. Restore file authentication on the primary cluster based on the information saved in step 1.

Restore steps for authentication data:

Use the following steps on a protocol node in the primary cluster and the secondary cluster to restore the authentication configuration.

Note: The object authentication does not need to be removed and re-added as part of failover, failback, or restore.

1. Save the current file authentication information on the primary cluster.
2. Remove file authentication from the primary cluster.
3. Restore file authentication on the primary cluster based on the information saved in step 1.
4. Save the current file authentication information on the secondary cluster.
5. Remove file authentication from the secondary cluster.
6. Restore file authentication on the secondary cluster based on the information saved in step 4.

CES data required for protocols cluster DR

Cluster Export Services (CES) data required in case of a disaster recovery scenario is as follows.

Cluster Configuration Repository (CCR) files that need to be backed up for CES in a disaster recovery scenario are as follows:
- mmsdrfs
- cesiplist

Failover steps for CES

No Cluster Export Services (CES) configuration information is restored on fail over. This is because this information is typically cluster specific and it would interfere with the proper operating of the secondary cluster.

Failback or recovery steps for CES

Use the following steps on a protocol node in the primary cluster to fail back or recover the CES configuration.

1. Restore the `cesiplist` file.
2. For each protocol node listed in the stored, backup copy of the `mmsdrfs` file, verify that the node on the primary cluster is also configured as a protocol node. If not, use the **mmchnode** to enable the node as a protocol node.
3. For each of the following CES parameters in the stored, backup copy of the `mmsdrfs` file, verify that the value is the same on the primary cluster. If not, use the **mmchconfig** to update the configuration value.
   - **cesSharedRoot**
   - **cesAddressPool**
   - **cesServices**
   - **cifsBypassTraversalChecking**
   - **syncSambaMetadataOps**
   - **cifsBypassShareLocksOnRename**

Chapter 34. File Placement Optimizer

GPFS File Placement Optimizer (FPO) is a set of features that allow GPFS to operate efficiently in a system based on a shared nothing architecture. It is useful for big data applications that process massive amounts of data.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

FPO uses the following entities and policies:

Chunks

A chunk is a logical grouping of blocks that allows the grouping to behave like one large block, useful for applications that need high sequential bandwidth. Chunks are specified by a block group factor that dictates how many file system blocks are laid out sequentially on disk to behave like a large block. Different Chunk size can be defined by block group factor on file level or defined globally on a storage pool by default.

On the file level, the block group factor can be specified by the **--block-group-factor** argument of the **mmchattr** command. You can also specify the block group factor by the **setBGF** argument of the **mmchpolicy** and **mmapplypolicy** command.The range of the block group factor is 1 - 1024. The default value is 1. You can also specify the block group factor through the **blockGroupFactor** argument in a storage pool stanza (as input to the **mmadddisk** or **mmcrfs** command).

The effective chunk size is a multiplication of Block Group Factor and GPFS block size. For example, setting block size to 1 MB and block group factor to 128 leads to an effective large block size of 128 MB.

See the following command descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmcrfs**
- **mmchpolicy**
- **mmapplypolicy**

Extended failure groups

A failure group is defined as a set of disks that share a common point of failure that might cause them all to become simultaneously unavailable. Traditionally, GPFS failure groups are identified by simple integers. In an FPO-enabled environment, a failure group might be specified as not just a single number, but as a vector of up to three comma-separated numbers. This vector conveys topology information that GPFS exploits when making data placement decisions.

In general, a topology vector is a way for the user to specify which disks are closer together and which are farther away. In practice, the three elements of the failure group topology vector might represent the rack number of a disk, a position within the rack, and a node number. For example, the topology vector 2,1,0 identifies rack 2, bottom half, first node.

Also, the first two elements of the failure group represent the failure group ID and the three elements together represent the locality group ID. For example, 2,1 is the failure group ID and 2,1,0 is the locality group ID for the topology vector 2,1,0.

The Data block placement decisions about the disk selection for data replica are made by GPFS based on the Failure group. When considering two disks for striping or replica placement purposes, it is important to understand the following:

- Disks that differ in the first of the three numbers are farthest apart (as they are in different racks).
- Disks that have the same first number but differ in the second number are closer (as they are in the same rack, but in different halves).
- Disks that differ only in the third number reside in different nodes in the same half of the same rack.
- Only disks that have all three numbers in common reside in the same node.

The data block placement decisions are also affected by the level of replication and the value of the **writeAffinityDepth** parameter. For example, when using replication 3, GPFS might place two replicas far apart (different racks) to minimize chances of losing both. However, the third replica can be placed close to one of the others (same rack, but different half), to reduce network traffic between racks when writing the three replicas.

To specify the topology vector that identifies a failure group, you use the **failureGroup=***FailureGroup* attribute in an NSD stanza (as input to the **mmadddisk** or **mmcrfs** command).

See the following command descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:
- **mmadddisk**
- **mmcrfs**

Write affinity depth

Write affinity depth is a policy that allows the application to determine the layout of a file in the cluster to optimize for typical access patterns. The write affinity is specified by a depth that indicates the number of localized copies (as opposed to wide striped). It can be specified at the storage pool or file level. The enabling of Write affinity depth, indicates that the first replica is being written on the node where the writing is triggered. It also indicates, the second and third replica (if any) are being written on the other node disks.

To specify write affinity depth, you use the **writeAffinityDepth** attribute in a storage pool stanza (as input to the **mmadddisk** or **mmcrfs** command) or the **--write-affinity-depth** argument of the **mmchattr** command. You can use **--block-group-factor** argument of the **mmchpool** command to change a storage pool's block group factor. You can change write affinity depth by **--write-affinity-depth** argument of **mmchpool** for a storage pool. You can also specify the write affinity depth for file by the `setWAD` argument of the **mmchpolicy** and **mmapplypolicy** commands.

A write affinity depth of 0 indicates that each replica is to be striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. By default, the unit of striping is a block; however, if the block group factor is specified in order to exploit chunks, the unit of striping is a chunk.

A write affinity depth of 1 indicates that the first copy is written to the writer node. The second copy is written to a different rack. The third copy is written to the same rack as the second copy, but on a different half (which can be composed of several nodes).

A write affinity depth of 2 indicates that the first copy is written to the writer node. The second copy is written to the same rack as the first copy, but on a different half (which can be composed of several nodes). The target node is determined by a hash value on the fileset ID of the file, or it is chosen randomly if the file does not belong to any fileset. The third copy is striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. The following conditions must be met while using a write affinity depth of 2 to get evenly allocated space in all disks:
1. The configuration in disk number, disk size, and node number for each rack must be similar.
2. The number of nodes must be the same in the bottom half and the top half of each rack.

This behavior can be altered on an individual file basis by using the **--write-affinity-failure-group** option of the **mmchattr** command.

Note: In fileset level, Write affinity depth of 2 is design to assign (write) all the files in a fileset to the same second-replica node. However, this behavior depends on node status in the cluster. After a node is added to or deleted from a cluster, a different node might be selected as the second replica for files in a fileset.

See the description of storage pool stanzas that follows. Also, see the following command descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmcrfs**
- **mmchpolicy**
- **mmapplypolicy**
- **mmchpool**

Write affinity failure group

Write affinity failure group is a policy that indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in a particular file are to be written. The policy allows the application to determine the layout of a file in the cluster to optimize for typical access patterns.

You specify the write affinity failure group through the **write-affinity-failure-group** *WafgValueString* attribute of the **mmchattr** command. You can also specify write affinity failure group through the **setWADFG** attribute of the **mmchpolicy** and **mmapplypolicy** command. Failure group topology vector ranges specify the nodes, and the specification is repeated for each replica of the blocks in a file.

For example, the attribute 1,1,1:2;2,1,1:2;2,0,3:4 indicates:

- The first replica is on rack 1, rack location 1, nodes 1 or 2.
- The second replica is on rack 2, rack location 1, nodes 1 or 2.
- The third replica is on rack 2, rack location 0, nodes 3 or 4.

The default policy is a null specification. This default policy indicates that each replica must follow the storage pool or the file-write affinity depth (WAD) definition for data placement. Not wide striped over all disks.

When data in an FPO pool is backed up in a IBM Spectrum Protect server and then restored, the original placement map is broken unless you set the write affinity failure group for each file before backup.

Note: To change the failure group of a disk in a write-affinity–enabled storage pool, you must use the **mmdeldisk** and **mmadddisk** commands. You cannot use **mmchdisk** to change it directly.

See the following command descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:

- **mmchpolicy**
- **mmapplypolicy**
- **mmchattr**

Enabling the FPO features

To efficiently support write affinity and the rest of the FPO features, GPFS internally requires the creation of special allocation map formats. When you create a storage pool that is to contain files that make use of FPO features, you must specify **allowWriteAffinity=yes** in the storage pool stanza.

To enable the policy to read from preferred replicas, issue one of the following commands:

- To specify that the policy read from the first replica, regardless of whether there is a replica on the disk, default to or issue the following:

    `mmchconfig readReplicaPolicy=default`

- To specify that the policy read replicas from the local disk, if the local disk has data, issue the following:

    `mmchconfig readReplicaPolicy=local`

- To specify that the policy read replicas from the fastest disk to read from based on the disk's read I/O statistics, run the following:

    `mmchconfig readReplicaPolicy=fastest`

Note: In an FPO-enabled file system, if you run data locality awareness workload over FPO, such as Hadoop or Spark, configure **readReplicaPolicy** as *local* to read data from the local disks to reduce the network bandwidth consumption.

See the description of storage pool stanzas that follows. Also, see the following command descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:

- **mmadddisk**
- **mmchconfig**
- **mmcrfs**

Storage pool stanzas

Storage pool stanzas are used to specify the type of layout map and write affinity depth, and to enable write affinity, for each storage pool.

Storage pool stanzas have the following format:

```
%pool:
  pool=StoragePoolName
  blockSize=BlockSize
  usage={dataOnly | metadataOnly | dataAndMetadata}
  layoutMap={scatter | cluster}
  allowWriteAffinity={yes | no}
  writeAffinityDepth={0 | 1 | 2}
  blockGroupFactor=BlockGroupFactor
```

See the following command descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:

- **mmadddisk**
- **mmcrfs**
- **mmchpool**

Recovery from disk failure

A typical shared nothing cluster is built with nodes that have direct-attached disks. Disks are not shared between nodes as in a regular GPFS cluster, so if the node is inaccessible, its disks are also inaccessible. GPFS provides means for automatic recovery from these and similar common disk failure situations.

The following command sets up and activates the disk recovery features:

`mmchconfig restripeOnDiskFailure=yes -i`

Whether a file system went through a recovery is determined by the max replication values for the file system. If the **mmlsfs -M** or **-R** value is greater than one, then the recovery code is run. The recovery actions are asynchronous and GPFS continues its processing while the recovery attempts take place. The results from the recovery actions and any errors that are encountered are recorded in the GPFS logs.

Two more parameters are available for fine-tuning the recovery process:

```
mmchconfig metadataDiskWaitTimeForRecovery=seconds
mmchconfig dataDiskWaitTimeForRecovery=seconds
```

The default value for **metadataDiskWaitTimeForRecovery** is 1800 seconds. The default value for **dataDiskWaitTimeForRecovery** is 3600 seconds.

See the following command description in the *IBM Spectrum Scale: Command and Programming Reference*:
- **mmchconfig**

Distributing data across a cluster

You can distribute data uniformly across a cluster.

Following are the possible ways to distribute the data:
- Import the data through a node that does not have any attached NSD and takes the role as a GPFS client node in the cluster. This ensures that the data is distributed evenly across all failure groups and all nodes within a failure group.
- Use a write affinity depth of 0 across the cluster.
- Make every GPFS node an ingest node and deliver data equally across all ingest nodes. However, this strategy is expensive in terms of implementation.

Ideally, all the failure groups must have an equal number of disks with roughly equal capacity. If one failure group is much smaller than the rest, it is likely to fill up faster than the others, and this complicates rebalancing actions.

After the initial ingesting of data, the cluster might be unbalanced. In such a situation, use the **mmrestripefs** command with the **-b** option to rebalance the data.

Note: For FPO users, the **mmrestripefs -b** command breaks the original data placement that follows the data locality rule.

FPO pool file placement and AFM

For AFM home or cache, an FPO pool file that is written on the local side is placed according to the write affinity depth and write affinity failure group definitions of the local side.

When a file is synced from home to cache, it follows the same FPO placement rule as when written from the gateway node in the cache cluster. When a file is synced from cache to home, it follows the same FPO data placement rule as when written from the NFS server in the home cluster.

To retain the same file placement at AFM home and cache, ensure that each has the same cluster configuration and set the write affinity failure group for each file. If the home and cache cluster have different configurations, such as the disk number, node number, or fail group, then the data locality might be broken.

Configuring FPO

Follow the steps listed in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* to install the GPFS RPMs and build the portability layer on all nodes in the cluster. You can configure password-less SSH for root user across all GPFS nodes. However, in cases of special security control, you can configure at least one node for the root user to access all GPFS nodes in a password-less mode. GPFS commands can be run only over these nodes.

For OS with Linux kernel 2.6, enter following commands on all GPFS nodes that are set as root to set vm.min_free_bytes :

```
# TOTAL_MEM=$(cat /proc/meminfo | grep MemTotal | tr -d \"[:alpha:]\" | tr -d \"[:punct:]\" | tr
-d \"[:blank:]\") # VM_MIN_FREE_KB=$((${TOTAL_MEM}*6/100))
```

```
# echo "vm.min_free_kbytes = $VM_MIN_FREE_KB" >> /etc/sysctl.conf # sysctl -p
# sysctl -a | grep vm.min_free_kbytes
```

Configuring IBM Spectrum Scale Clusters

All GPFS configuration steps must be performed as the root user. These steps need to be executed only on one node, not on all nodes.

Create the GPFS Cluster

The GPFS node file defines all nodes in the cluster and some of the roles.

Create the GPFS cluster with **node11** as the primary and **node21** as the secondary cluster configuration server. Set the *-A* flag to automatically start GPFS daemons when the OS is started.

```
# mmcrcluster -A -C gpfs-cluster -p node11 -s node21 -N nodefile -r $(which ssh) -R $(which scp)
```

Use the `mmlscluster` command to view the cluster.

Apply IBM Spectrum Scale license

All GPFS nodes require a license designation before they can be used. The FPO feature introduced a dedicated PFS license class **fpo**. In a GPFS FPO cluster, all quorum and manager nodes require a server license. Based on the sample environment, **node11**, **node21**, and **node31** require a server license. The other nodes require an **fpo** license.

```
# mmchlicense server --accept -N node11,node21,node31
```

```
# mmchlicense fpo --accept -N node12,node13,node14,node15,node16,node22,node23,node24,node25,node26,
node32,node33,node34,node35,node36
```

Use the `mmlslicense -L` command to view license information for the cluster.

Nodes with no disks in the file system are called as diskless nodes. Run the **mmchlicense client --accept -N** command to accept the client license for disks that have no disks in the GPFS file system.

Start the GPFS cluster to verify whether it starts successfully. Use the **mmstartup** *-a* command to start the GPFS cluster and the **mmgetstate** *-a* command to view the state of the GPFS cluster.

Create GPFS Network Shared Disks (NSD)

To create the network shared disks (NSD) in GPFS, create a disk file to be used as input to the `mmcrnsd` command. The disk file defines the GPFS pools and the NSDs. A recommended GPFS pool configuration has two storage pools, a system pool for metadata only and a data pool.

- Storage Pools
  - System pool contains all of the metadata disks and does not have FPO behavior enabled. The system pool should have a smaller block size than the data pool for performance reasons. If you choose to use `dataAndMetadata` disks in the system pool, you must set the system pool block size to be the same as the data pool block size as both the pools can have data. For the `dataAndMetadata` system pool, the block size 1M is recommended.
  - Data pool contains all of the data disks and has FPO behavior enabled by setting **allowWriteAffinity**=yes, **writeAffinityDepth**=1, and **blockGroupFactor**=128.

    The chunk size can be calculated as **blockSize * blockGroupFactor**. Similar to the HDFS recommendation, the GPFS FPO recommendation is `blockSize=2M * blockGroupFactor=64` for a chunk size of 128 MB
- NSD
  - Every local disk to be used by GPFS must have a matching entry in the disk stanza file
  - The device must match the device path of the disk.

    Note: In the example, /dev/sda is not included because this is the OS disk.

If **MapReduce** intermediate and temporary data is stored on etx3/ext4 disks instead of GPFS, make sure those disks are not included in the disk file or GPFS will format them and include them in the GPFS cluster

– System pool disks:

- Should have **usage=metadataOnly**. It is possible to use **usage=dataAndMetadata** if there is a reason to have data on the system pool disks. The block size of the dataAndMetadata system pool must be the same as the block size of a data pool in the file system.

- **failureGroup** must be a single number if **allowWriteAffinity** is not enabled (specify **allowWriteAffinity=no** for system pool definition when doing **mmcrnsd** or **mmcrfs**) and it should be the same for all disks on the same node. If **allowWriteAffinity** is enabled for system pool, the failure group can be of format *rack,position,node*, for example, **2,0,1**; or, it can take the traditional single-number failure group format also.

- Even when **allowWriteAffinity** is enabled for system pool, the metadata does not follow data locality rules; these rules apply only to data placement

– Data pool disks:

- Must have **usage**=dataOnly.

- **failureGroup** must be of the format [rack,position,node], where position is either 0 or 1 to represent top or bottom half of the rack. The sample environment does not have half racks, so the same position is used for all nodes. Especially, when position and node fields are ignored in the cluster, the failure group can be defined as a single number, [rack, -,-].

Example of NSD disk file created by using the **mmcrnsd** command:

```
%pool: pool=system  blockSize=256K layoutMap=cluster allowWriteAffinity=no
%pool: pool=datapool blockSize=2M layoutMap=cluster allowWriteAffinity=yes writeAffinityDepth=1
blockGroupFactor=256

# gpfstest9
%nsd: nsd=node9_meta_sdb  device=/dev/sdb servers=gpfstest9 usage=metadataOnly failureGroup=1 pool=system

%nsd: nsd=node9_data_sdf2  device=/dev/sdf servers=gpfstest9 usage=dataOnly failureGroup=1,0,1 pool=datapool
%nsd: nsd=node9_data_sdg2  device=/dev/sdg servers=gpfstest9 usage=dataOnly failureGroup=1,0,1 pool=datapool

#gpfstest10
%nsd: nsd=node10_meta_sda  device=/dev/sda servers=gpfstest10 usage=metadataOnly failureGroup=2 pool=system

%nsd: nsd=node10_data_sde2  device=/dev/sde servers=gpfstest10 usage=dataOnly failureGroup=2,0,1 pool=datapool
%nsd: nsd=node10_data_sdg2  device=/dev/sdg servers=gpfstest10 usage=dataOnly failureGroup=2,0,1 pool=datapool

#gpfstest11

%nsd: nsd=node11_meta_sdb  device=/dev/sdb servers=gpfstest11 usage=metadataOnly failureGroup=3 pool=system

%nsd: nsd=node11_data_sdf2  device=/dev/sdf servers=gpfstest11 usage=dataOnly failureGroup=3,0,1 pool=datapool
%nsd: nsd=node11_data_sdg2  device=/dev/sdg servers=gpfstest11 usage=dataOnly failureGroup=3,0,1 pool=datapool
```

If any disks are previously used by GPFS, you must use the **-v no** flag to force GPFS to use them again.

Note: Use the **-v no** flag only if you are sure that the disk can be used by GPFS.
Use the # mmcrnsd -F diskfile [-v no] command to create NSDs and use the **mmlsnsd –m** command to display the NSDs.

Apply GPFS FPO configuration changes
GPFS FPO requires several global GPFS configuration changes to operate successfully.

Set the GPFS page pool to 25% of system memory on each node. For Hadoop noSQL application, the page pool of GPFS FPO can be configured for better performance, for example, 30% of physical memory.

In this example, all nodes have the same amount of memory, which is a best practice. If some nodes have different memory, set the page pool on a per-node basis by using the -N flag.

```
# TOTAL_MEM=$(cat /proc/meminfo | grep MemTotal | tr -d \"[:alpha:]\" | tr -d\"[:punct:]\" | tr -d \"[:blank:]\")
# PAGE_POOL=$(((${TOTAL_MEM}*25/(100*1024)))
# mmchconfig pagepool=${PAGE_POOL}M
```

Start the GPFS cluster:
```
# mmstartup -a
# mmgetstate -a
```

Use the **mmlsconfig** and **mmdiag** commands to see the configuration changes:
```
# mmlsconfig
# mmdiag --config
```

Create the GPFS file system and pools

After you create NSDs, a GPFS file system can be created.

To use FPO, a single file system is recommended. The following example creates a file system with mount point **/mnt/gpfs** that is set to auto mount. This mount point is used in Hadoop configuration later. The replication for both data and metadata is set to 3 replicas. Quotas are not activated on this file system. An inode size of 4096 is recommended for typical **MapReduce** data sizes \ **-S** and **-E** settings help improve performance for **mtime** and **atime** updates. The **mmcrfs** command also creates GPFS storage pools based on the disk file %pools setting.
```
# mmcrfs gpfs-fpo-fs -F diskfile -T /mnt/gpfs -n 32 -m 3 -M 3 -r 3 -R 3 -i 4096 -A yes -Q no -S relatime -E no [-v no]
```

For more information on the pool configuration, see "Create GPFS Network Shared Disks (NSD)" on page 512.

Mount the file system on all nodes:
```
# mmmount all -a
```

Use the **mmlsfs** command to display the file system configuration:
```
# mmlsfs all
```

Use the **mmlsdisk** command to display the status of the NSDs:
```
# mmlsdisk gpfs-fpo-fs —L
```

Use the **mmdf** command to view the disk usage for the file system:
```
# mmdf gpfs-fpo-fs
```

Use the **mmlspool** command to view the storage pools:
```
# mmlspool gpfs-fpo-fs all —L
```

Create GPFS Data Placement Policy

Before data can be written to a GPFS file system that has more than one pool, you must apply a data placement policy.

In this example, all of the data goes to data pool.
```
# cat policyfile
rule default SET POOL 'datapool'
```

After you create the rule file, use the **mmchpolicy** command to enable the policy:
```
# mmchpolicy gpfs-fpo-fs policyfile -I yes
```

Use the **mmlspolicy** command to display the currently active rule definition:
```
# mmlspolicy gpfs-fpo-fs —L
```

Create filesets for `MapReduce` intermediate and temporary data

To efficiently store MapReduce intermediate and temporary data, use GPFS filesets and policies to better emulate local disk behavior.

Note: If **MapReduce** intermediate and temporary data is not stored on GPFS, `mapred.cluster.local.dir` in MRv1 or `yarn.nodemanager.log-dirs` and `yarn.nodemanager.local-dirs` in Hadoop Yarn does not point to a GPFS directory, you do not need to go through this section.

Create an independent fileset

Consider using `--inode-space new [--inode-limit MaxNumInodes[:NumInodesToPreallocate]` to create an independent fileset. This can improve the performance for the fileset but requires calculation for `MaxNumInodes` and `NumInodesToPreallocate`. `MaxNumInodes` must be eight times the number of files expected on the fileset, and `NumInodesToPreallocate` must be half the value of `MaxNumInodes`. See the `mmcrfileset` man page to understand this option.

Use the `mmcrfileset` command to create two filesets, one for local intermediate data and one for temporary data:
```
# mmcrfileset gpfs-fpo-fs mapred-local-fileset
# mmcrfileset gpfs-fpo-fs mapred-tmp-fileset
```

After the fileset is created, it must be linked to a directory under this GPFS file system mount point. This example uses **/mnt/gpfs/mapred/local** for intermediate data and **/mnt/gpfs/tmp** for temporary data. As **/mnt/gpfs/mapred/local** is a nested directory, the directory structure must exist before linking the fileset. These two directories are required for configuring Hadoop.
```
# mkdir -p $(dirname /mnt/gpfs/mapred/local)
# mmlinkfileset gpfs-fpo-fs mapred-local-fileset -J /mnt/gpfs/mapred/local
# mmlinkfileset gpfs-fpo-fs mapred-tmp-fileset -J /mnt/gpfs/tmp
```

Use the `mmlsfileset` command to display fileset information:
```
# mmlsfileset gpfs-fpo-fs -L
```

The next step to setting up the filesets is to apply a GPFS policy so the filesets act like local directories on each node. This policy instructs GPFS not to replicate the data for these two filesets, and since these filesets are stored on the data pool, they can use FPO features that keeps local writes on local disks. Metadata must still be replicated three times, which can result in performance overhead. File placement policies are evaluated in the order they are entered, so ensure that the policies for the filesets appear before the default rule.
```
# cat policyfile
rule 'R1' SET POOL 'datapool' REPLICATE (1,1) FOR FILESET ('mapred-local-fileset')
rule 'R2' SET POOL 'datapool' REPLICATE (1,1) FOR FILESET ('mapred-tmp-fileset')
rule default SET POOL 'datapool'
# mmchpolicy gpfs-fpo-fs policyfile -I yes
```

Use the `mmlspolicy` command to display the currently active rule definition:
```
# mmlspolicy gpfs-fpo-fs —L
```

In each of these filesets, create a subdirectory for each node that run Hadoop jobs. Based on the sample environment, this script creates these subdirectories:
```
# cat mk_gpfs_local_dirs.sh
#!/bin/sh for nodename in $(mmlsnode -N all); do
 mkdir -p /mnt/gpfs/tmp/${nodename}
 mkdir -p /mnt/gpfs/mapred/local/${nodename}
```

```
done
```

After that, on **${nodename}, link /mnt/gpfs/tmp/${nodename} /hadoop/tmp; link /mnt/gpfs/mapred/ local/${nodename} /hadoop/local**. Then, in Hadoop cluster, configure **/hadoop/tmp** as **hadoop.tmp.dir** in all Hadoop nodes; configure **/hadoop/local** as **mapred.cluster.local.dir** in MRv1 or **yarn.nodemanager.log-dirs** and **yarn.nodemanager.local-dirs** in Hadoop Yarn for Hadoop nodes.

To check that the rules are working properly, you can write some test files and verify their replication settings. For example:

Create some files:
```
# echo "test" > /mnt/gpfs/mapred/local/testRep1
# echo "test" > /mnt/gpfs/testRep3
```

Use the **mmlsattr** command to check the replication settings
```
# mmlsattr /mnt/gpfs/mapred/local/testRep1
replication factors
metadata(max) data(max) file [flags]
 -------------------------------------
 1 ( 3) 1 ( 3) /mnt/gpfs/mapred/local/testRep1
 # mmlsattr /mnt/gpfs/testRep3
replication factors
metadata(max) data(max) file [flags]
-------------------------------------
3 ( 3) 3 ( 3) /mnt/gpfs/testRep3
```

Set file system permissions
Depending on how different users interact with GPFS, you must create a user directory with permissions that allow users to create their own home directories.
```
# mkdir -p /mnt/gpfs/user
# chmod 1777 /mnt/gpfs/user
```

To make sure that MapReduce jobs can write to the GPFS file system, assign permissions to the CLUSTERADMIN user. CLUSTERADMIN is the user who starts Hadoop **namenode** and **datanode** service, for example, user hdfs.
```
# chown -R CLUSTERADMIN:CLUSTERADMINGROUP /mnt/gpfs
# chmod -R +rx /mnt/gpfs
```

Use the **ls** command to verify the permission settings:
```
# ls -lR /mnt/gpfs
```

Basic Configuration Recommendations

Operating system configuration and tuning

Perform the following steps to configure and tune a Linux system:
1. **deadline disk scheduler**

   Change all the disks defined to IBM Spectrum Scale to use the 'deadline' queue scheduler (cfg is the default for some distros, such as RHEL 6).

   For each block device defined to IBM Spectrum Scale, run the following command to enable the deadline scheduler:
   ```
   echo "deadline" > /sys/block/<diskname>/queue/scheduler
   ```

Changes made in this manner (echoing changes to sysfs) do not persist over reboots. To make these changes permanent, enable the changes in a script that runs on every boot or (generally preferred) create a udev rule.

The following sample script sets deadline scheduler for all disks in the cluster that are defined to IBM Spectrum Scale (this example must be run on the node with passwordless access to all the other nodes):

```
#!/bin/bash
/usr/lpp/mmfs/bin/mmlsnsd -X | /bin/awk ' { print $3 " " $5 } ' | \
/bin/grep dev |
while read device node ; do
device=$(echo $device | /bin/sed 's/\/dev\///' )
/usr/lpp/mmfs/bin/mmdsh -N $node "echo deadline >
/sys/block/$device/queue/scheduler"
Done
```

As previously stated, changes made by echoing to sysfs files (as per this example script) take effect immediately on running the script, but do not persist over reboots. One approach to making such changes permanent is to enable a udev rule, as per this example rule to force all block devices to use deadline scheduler after rebooting. To enable this rule, you can create the following file as /etc/udev/rules.d/99-hdd.rules):

```
ACTION=="add|change", SUBSYSTEM=="block", ATTR{device/model}=="*",
ATTR{queue/scheduler}="deadline"
```

In the next step, give an example of how to create udev rules that apply only to the devices used by IBM Spectrum Scale.

2. **disk IO parameter change**

   To further tune the block devices used by IBM Spectrum Scale, run the following commands from the console on each node:

   ```
   echo 16384 > /sys/block/<device>/queue/max_sectors_kb
   echo 256 > /sys/block/<device>/queue/nr_requests
   echo 32 > /sys/block/<device>/device/queue_depth
   ```

   These block device tuning settings must be large enough for SAS/SATA disks. For /sys/block/<device>/queue/max_sectors_kb, the tuning value chosen must be less than or equal to /sys/block/<device>/queue/max_hw_sectors_kb. Many SAS/SATA devices allow setting 16384 for *max_hw_sectors_kb*, but not all devices may accept these values. If your device does not allow for the block device tuning recommendations above, try setting smaller values and cutting the recommendations in half until the tuning is successful. For example, if setting *max_sectors_kb* to 16384 results in a write error:

   ```
   echo 16384 > /sys/block/sdd/queue/max_sectors_kb
   -bash: echo: write error: Invalid argument
   ```

   Try setting *max_sectors_kb* to 8192:

   ```
   echo 8192 > /sys/block/sdd/queue/max_sectors_kb
   ```

   If your disk is not SAS/SATA, check the disk specification from the disk vendor for tuning recommendations.

   Note: If the max_sectors_kb of your disks is small (e.g. 256 or 512) and you are not allowed to tune the above values (i.e., you get an "invalid argument" as per the example above), then your disk performance might be impacted because IBM Spectrum Scale IO requests might be split into several smaller requests according to the limits max_sectors_kb places at the block device level.

   As discussed in Step 1 tuning recommendations, any tuning done by echoing to sysfs files will be lost when a node reboots. To make such a tuning permanent, either create appropriate udev rules or place these commands in a boot file that is run on each reboot.

   As udev rules are the preferred way of accomplishing this kind of block device tuning, give an example of a generic udev rule that enables the block device tuning recommended in steps 1 and 2 for all block devices. This rule can be enabled by creating the following rule as a file /etc/udev/rules.d/100-hdd.rules):

```
ACTION=="add|change", SUBSYSTEM=="block", ATTR{device/model}=="*",
ATTR{queue/nr_requests}="256", ATTR{device/queue_depth}="32",
ATTR{queue/max_sectors_kb}="16384"
```

If it is not desirable to tune all block devices with the same settings, multiple rules can be created with specific tuning for the appropriate devices. To create such device specific rules, you can use the 'KERNEL' match key to limit which devices udev rules apply to (e.g., KERNEL==sdb). The following example script can be used to create udev rules that tune only the block devices used by IBM Spectrum Scale:

```
#!/bin/bash
#clean up any existing /etc/udev/rules.d/99-hdd.rules files
/usr/lpp/mmfs/bin/mmdsh -N All "rm -f /etc/udev/rules.d/100-hdd.rules"
#collect all disks in use by GPFS and create udev rules one disk at a time
/usr/lpp/mmfs/bin/mmlsnsd -X | /bin/awk ' { print $3 " " $5 } ' | \
/bin/grep dev |
while read device node ; do
device=$(echo $device | /bin/sed 's/\/dev\///' )
echo $device $node
echo "ACTION==\"add|change\", SUBSYSTEM==\"block\", \
KERNEL==\"$device\", ATTR{device/model}==\"*\", \
ATTR{queue/nr_requests}=\"256\", \
ATTR{device/queue_depth}=\"32\", ATTR{queue/max_sectors_kb}=\"16384\" "> \
/tmp/100-hdd.rules
/usr/bin/scp /tmp/100-hdd.rules $node:/tmp/100-hdd.rules
/usr/lpp/mmfs/bin/mmdsh -N $node "cat /tmp/100-hdd.rules >>\
/etc/udev/rules.d/100-hdd.rules"
Done
```

Note: The previous example script must be run from a node that has ssh access to all nodes in the cluster. This previous example script will create udev rules that will set the recommended block device tuning on future reboots. To put the recommended tuning values from steps 1 and 2 in place immediately in effect, the following example script can be used:

```
#!/bin/bash
/usr/lpp/mmfs/bin/mmlsnsd -X | /bin/awk ' { print $3 " " $5 } ' | \
/bin/grep dev |
while read device node ; do
device=$(echo $device | /bin/sed 's/\/dev\///' )
/usr/lpp/mmfs/bin/mmdsh -N $node "echo deadline >\
/sys/block/$device/queue/scheduler"
/usr/lpp/mmfs/bin/mmdsh -N $node "echo 16384>\
/sys/block/$device/queue/max_sectors_kb"
/usr/lpp/mmfs/bin/mmdsh -N $node "echo 256 >\
/sys/block/$device/queue/nr_requests"
/usr/lpp/mmfs/bin/mmdsh -N $node "echo 32 >\
/sys/block/$device/device/queue_depth"
Done
```

3. **disk cache checking**

   On clusters that do not run Hadoop/Spark workloads, disks used by IBM Spectrum Scale must have physical disk write caching disabled, regardless of whether RAID adapters are used for these disks.

   When running other (non-Hadoop/Spark) workloads, write caching on the RAID adapters can be enabled if the local RAID adapter cache is battery protected, but the write cache on the physical disks must not be enabled.

   Check the specification for your RAID adapter to figure out how to turn on/off the RAID adapter write cache, as well as the physical disk write cache.

   For common SAS/SATA disks without RAID adapter, run the following command to check whether the disk in question is enabled with physical disk write cache:

   ```
   sdparm --long /dev/<diskname> | grep WCE
   ```

   If WCE is 1, it means the disk write cache is on.

   The following commands can be used to turn on/off physical disk write caching:

```
# turn on physical disk cache
sdparm -S -s WCE=1 /dev/<diskname>
# turn off physical disk cache
sdparm -S -s WCE=0 /dev/<diskname>
```

Note: The physical disk read cache must be enabled no matter what kind of disk is used. For SAS/SATA disks without RAID adapters, run the following command to check whether the disk read cache is enabled or not:

```
sdparm --long /dev/<diskname> | grep RCD
```

If the value of RCD (Read Cache Disable) is 0, the physical disk read cache is enabled. On Linux, usually the physical disk read cache is enabled by default.

4. **Tune vm.min_free_kbytes to avoid potential memory exhaustion problems.**

   When vm.min_free_kbytes is set to its default value, in some configurations it is possible to encounter memory exhaustion symptoms when free memory must be available. It is recommended that vm.min_free_kbytes be set to between 5~6 percent of the total amount of physical memory, but no more than 2GB should be allocated for this reserve memory.

   To tune this value, add the following into /etc/sysctl.conf and then run 'sysctl -p' on Red Hat or SuSE:

   ```
   vm.min_free_kbytes = <your-min-free-KBmemory>
   ```

5. **OS network tuning**

   If your network adapter is 10Gb Ethernet adapter, you can put the following into /etc/sysctl.conf and then run **/sbin/sysctl -p /etc/sysctl.conf** on each node:

   ```
   sunrpc.udp_slot_table_entries = 128
   sunrpc.tcp_slot_table_entries = 128
   net.core.rmem_max=4194304
   net.core.wmem_max=4194304
   net.core.rmem_default=4194304
   net.core.wmem_default=4194304
   net.core.netdev_max_backlog = 300000
   net.core.somaxconn = 10000
   net.ipv4.tcp_rmem = 4096 4224000 16777216
   net.ipv4.tcp_wmem = 4096 4224000 16777216
   net.core.optmem_max=4194304
   ```

   If your cluster is based on InfiniBand adapters, see the guide from your InfiniBand adapter vendor.

   If you bond two adapters and configure xmit_hash_policy=layer3+4 with bond mode 4 (802.3ad, the recommended bond mode), IBM Spectrum Scale of one node has only one TCP/IP connection with another node in the cluster for data transfer. This might make the network traffic only over one physical connection if the network traffic is not heavy.

   If your cluster size is not large (for example, only one physical switch is enough for your cluster nodes), you could try bonding mode 6 (balance-alb, no special support from switch). This might give better network bandwidth as compared with bonding mode 4 (802.3ad, require support from switch). See the Linux bonding: 802.3ad (LACP) vs. balance-alb mode link for advantages and disadvantages on Linux bonding 802.3ad versus balance-alb mode.

IBM Spectrum Scale configuration and tuning

Perform the following steps to tune the IBM Spectrum Scale cluster and file systems:

1. **Data replica and metadata replica:**

   While creating IBM Spectrum Scale file systems, ensure that the replication settings meet the data protection needs of the cluster.

   For production cluster over internal disks, it is recommended to take replica 3 for both data and metadata. If you have local RAID5 or RAID6 adapters with battery protected, you can take replica 2 for the data.

   When a file system is created, the default number of copies of data and metadata are respectively defined by the **-r** (DefaultDataReplicas) and **-m** (DefaultMetadataReplicas) options to the **mmcrfs**

command. Also, the value of `-R` (MaxDataReplicas) and `-M` (MaxMetadataReplicas) cannot be changed after the file system is created. Therefore, it is recommended to take 3 for `-R`/`-M` for flexibility to change the replica in the future.

Note: The first instance (copy) of the data is referred to as the first replica. For example, setting the `DefaultDataReplicas=1` (via `'-d 1'` option to **mmcrfs**) results in only a single copy of each piece of data, which is typically not desirable for a shared-nothing environment.

Query the number of replicas kept for any given file system by running the command:

```
/usr/lpp/mmfs/bin/mmlsfs <filesystem_name> | egrep " -r| -m"
```

Change the level of data and metadata replication for any file system by running **mmchfs** by using the same `-r` (DefaultDataReplicas) and `-m` (DefaultMetadataReplicas) flags to change the default replication options and then **mmrestripefs** (with the `-R` flag) to restripe the file system to match the new default replication options.

For example:

```
/usr/lpp/mmfs/bin/mmchfs <filesystem_name> -r <NewDefaultDataReplicas> -m
<NewDefaultDataReplicas>
/usr/lpp/mmfs/bin/mmrestripefs <filesystem_name> -R
```

2. **Additional considerations for the file system:**

When you create the file system, consider tuning the `/usr/lpp/mmfs/bin/mmcrfsparameters` file based on the characteristics of your applications:

- **-L**   By default, the value is 4MB for a file system log file. It is a good idea to create any file system with at least a 16 MB log file (`-L` 16M) or, if your application is sensitive to meta-operations, with at least a 32MB log file (`-L` 32M).

- **-E**   By default, the value is **yes**, which provides exact mtime. If your applications do not require exact mtime, you can change this value to **no** for better performance.

- **-S**   The default value depends on the minimum release level of the cluster when the file system is created. If the minimum release level is 5.0.0 or greater, the default value is **relatime**. Otherwise, the default value is **no**, which causes the atime to be updated each time that the file is read. If your application does not depend on exact atime, **yes** or **relatime** provides better performance.

--inode-limit
If you plan for the file system to contain a large number of files, it is a good idea to set the value as large as possible to avoid getting errors that say "no inode space". You can estimate the value of this parameter with the following formula:

--inode-limit = (*<metadata_disk_size>* * *<metadata_disk_number>*)/(*<inode_size>* * **DefaultMetadataReplicas**)

For more information, see the topics *mmchfs command* and *mmcrfs command* in the *IBM Spectrum Scale: Command and Programming Reference*.

3. **Define the data and the metadata distribution across the NSD server nodes in the cluster:**

Ensure that clusters larger than 4 nodes are not defined with a single (dataAndMetadata) system storage pool.

For performance and RAS reasons, it is recommended that data and metadata be separated in some configurations (which means that not all the storage is defined to use a single dataAndMetadata system pool).

These guidelines focus on the RAS considerations related to the implications of losing metadata servers from the cluster. In IBM Spectrum Scale Shared Nothing configurations (which recommend setting the `unmountOnDiskFail=meta` option), a given file system is unmounted when the number of nodes experiencing metadata disk failures is equal to or greater than the value of the `DefaultMetadataReplicas` option defined for the file system (the `-m` option to the **mmcrfs** command as per above). So, for a file system with the typically configured value `DefaultMetadataReplicas=3`, the file system will unmount if metadata disks in three separate locality group IDs fail (when a node fails, all the internal disks in that node will be marked down).

Note: All the disks in the same file system on a given node must have the same locality group ID. The Locality ID refers to all three elements of the extended failure group topology vector (For example, the vector 2,1,3 could represent rack 2, rack position 1, node 3 in this portion of the rack). To avoid file system unmounts associated with losing too many nodes serving metadata, it is recommended that the number of metadata servers be limited when possible. Also metadata servers must be distributed evenly across the cluster to avoid the case of a single hardware failure (such as the loss of a frame/rack or network switch) leading to multiple metadata node failures.

Some suggestions for separation of data and metadata based on cluster size:

Total NSD Nodes in Cluster	Suggestions for Allocation of Data and Metadata Across NSDs
1-5	All nodes must have both data and metadata disks. Depending on the available disks it is optional: both the data and metadata can be stored on each disk in this configuration (in which case, the NSDs will all be defined as dataAndMetadata) or the disks can be specifically allocated for data or metadata. If the disk number per node is equal to or less than 3, define all the disks as dataAndMetadata. If the disk number per node is larger than 3, take the 1:3 ratio for metadataOnly disk and dataOnly disk if your applications are meta data IO sensitive. If your applications are not metadata IO sensitive, consider using 1 metadataOnly disk.
6-9	5 nodes must serve as metadata disks. Assign one node per virtual rack where each node is one unique failure group. Among these nodes, select 5 nodes with metadata disks and other nodes with data-only disks. For metadata disk number, if you are not considering IOPS for metadata disk, you can select one disk as metadata NSD from the above 5 nodes with metadata disks. Other disks from these 5 nodes are used for data disks. if considering IOPS for metadata disk, you could select 1:3 ratio for metadata:data. For example, if you have 8 nodes with 10 disks per node, you have totally 81 disks. However, if you are considering 1:3 ratio, you could have 20 disks for metadata and select 5 disks per node from the above 5 nodes as metadata NSD disks. All other disks are configured as data NSD.
10 - 19	There are several different layouts, for example, 2 nodes per virtual rack for 10-node cluster; for 20-node cluster, you can take every 4 nodes per virtual rack or every 2 nodes per virtual rack; for 15-node cluster, you can take every 3 nodes per virtual rack. You must keep at least 5 failure groups for meta data and data. This can ensure that you have enough failure groups for data restripe when you have failures from 2 failure groups. To make it simple, it is suggested that every 2 nodes are defined as a virtual rack, with the first element of the extended failure group kept the same for nodes in the same virtual rack, and every virtual rack must have a node with metadata disks defined. For example, for an 18-node cluster, node1~node18, node1 and node2 are considered as a virtual rack. You can select some disks from node1 as metadataOnly disks and other disks from node1 and all disks from node2 as dataOnly disks. Ensure that these nodes are in the same failure group (for example, all dataOnly disks from node1 are of failure group 1,0,1, all dataOnly disks from node2 are of failure group 1,0,2).

Total NSD Nodes in Cluster	Suggestions for Allocation of Data and Metadata Across NSDs
20 or more	Usually, it is recommended that the virtual rack number be greater than 4 but less than 32 with each rack of the same node number. Each rack will be defined as one unique failure group and you will have 5+ failure groups that can tolerate failures from 2 failure groups for data restripe. Select one node from each rack to serve as meta data node. For example, for a 24-node cluster, you can split the clusters into 6 virtual racks with 4 nodes per rack. For 21-node cluster, it is recommended to take 7 virtual racks with 3 nodes per rack. For node number larger than 40, as a starting point, it's recommended that approximately every 10 nodes should be defined as a virtual rack, with the first element of the extended failure group kept the same for nodes in the same virtual rack. As for meta data, every virtual rack should have one node with metadataOnly disks defined. if you have more than 10+ racks, you could only select 5~10 virtual racks configured with metadata disks. As for how many disks must be configured as metadataOnly disks on the node which is selected for metadataOnly disks, this depends on the exact disk configuration and workloads. For example, if you configure one SSD per virtual rack, defining the SSD from each virtual rack as metadataOnly disks will work well for most workloads.

Note:

a. If you are not considering the IOPS requirement from the meta IO operations, usually 5% of the total disk size in the file system must be kept for meta data. If you can predict how many files your file system will be filled and the average file size, the requirement of the meta space size could be calculated roughly.

b. In a Shared Nothing framework, it is recommended that all nodes have similar disks in disk number and disk capacity size. If not, it might lead to hot disks when some nodes with small disk number or small disk capacity size are running out of disk space.

c. As for the number of nodes that are considered as one virtual rack, it is recommended to keep the node number even from each virtual rack.

d. It is always recommended to configure SSD or other fast disks as metadataOnly disks. This speeds up some maintenance operations, such as `mmrestripefs`, `mmdeldisk`, and `mmchdisk`.

e. If you are not sure about failure group definition, contact scale@us.ibm.com

4. When running a sharing nothing cluster, choose a failure group mapping scheme suited to IBM Spectrum Scale.

Defining more than 32 failure groups IDs for a given file system will slow down the execution of a lot of concurrent disk space allocation operations, such as restripe operations `mmrestripefs -b`.

On FPO-enabled clusters, defining more than 32 locality groups per failure group ID will slow down the execution of restripe operations, such as `mmrestripefs -r`.

To define an IBM Spectrum Scale FPO-enabled cluster containing a storage pool, set the option `allowWriteAffinity` to yes. This option can be checked by running the `mmlspool <fs-name> all - L` command. In FPO-enabled clusters, currently all disks on the same node must be assigned to the same locality group ID (three integer vector x,y,z), which also defines a failure group ID <x,y>. It is recommended that failure group IDs refer to sets of common resources, with nodes sharing a failure group ID having a common point of failure, such as a shared rack or a network switch.

5. Do not configure `allowWriteAffinity=yes` for a metadataOnly system pool.

For a metadataOnly storage pool (not a dataAndMetadata pool), set allowWriteAffinity to no. Setting allowWriteAffinity to yes for metadataOnly storage pool slows down the inode allocation for the pool.

6. Any FPO-enabled storage pool (any pool with allowWriteAffinity=yes defined) must define blockGroupFactor to be larger than 1 (regardless of the value of writeAffinityDepth).

When allowWriteAffinity is enabled, more RPC (Remote Procedure Call) activity might occur compared to the case of setting allowWriteAffinity=no.

To reduce some of the RPC overhead associated with setting `allowWriteAffinity=1`, for pools with allowWriteAffinity enabled, it is recommended that the BlockGroupFactor be set to greater than 1. Starting point recommendations are `blockGroupFactor=2` (for general workloads), `blockGroupFactor=10`(for database workloads), and `blockGroupFactor=128` (Hadoop workloads).

7. Tune the block size for storage pools defined to IBM Spectrum Scale.

   For storage pools containing both data and metadata (pools defined as dataAndMetadata), a block size of 1M is recommended.

   For storage pools containing only data (pools defined as dataOnly), a block size of 2M is recommended.

   For storage pools containing only metadata (pools defined as metadataOnly), a block size of 256K is recommended.

   The following sample pool stanzas (used when creating NSDs via the `mmcrnsd` command) are based on the tuning suggestions from steps 4-7:

   ```
   #for a metadata only system pool:
   %pool: pool=system blockSize=256K layoutMap=cluster allowWriteAffinity=no
   #for a data and metadata system pool:
   %pool: pool=system blockSize=1M layoutMap=cluster allowWriteAffinity=yes
   writeAffinityDepth=1 blockGroupFactor=2
   #for a data only pool:
   %pool: pool=datapool blockSize=2M layoutMap=cluster allowWriteAffinity=yes
   writeAffinityDepth=1 blockGroupFactor=10
   ```

8. Tune the size of the IBM Spectrum Scale **pagepool** attribute by setting the pool size of each node to be between 10% and 25% of the real memory installed.

   Note: The Linux buffer pool cache is not used for IBM Spectrum Scale file systems. The recommended size of the **pagepool** attribute depends on the workload and the expectations for improvements due to caching. A good starting point recommendation is somewhere between 10% and 25% of real memory. If machines with different amounts of memory are installed, use the `-N` option to `mmchconfig` to set different values according to the memory installed on the machines in the cluster. Though these are good starting points for performance recommendations, some customers use relatively small page pools, such as between 2-3% of real memory installed, particularly for machines with more than 256GB installed.

   The following example shows how to set a page pool size equal to 10% of the memory (this assumes all the nodes have the same amount of memory installed):

   ```
   TOTAL_MEM=$(cat /proc/meminfo | grep MemTotal | tr -d \"[:alpha:]\" | tr -d
   \"[:punct:]\" | tr -d \"[:blank:]\")
   PERCENT_OF_MEM=10
   PAGE_POOL=$((${TOTAL_MEM}*${PERCENT_OF_MEM}/(100*1024)))
   mmchconfig pagepool=${PAGE_POOL}M -i
   ```

9. Change the following IBM Spectrum Scale configuration options and then restart IBM Spectrum Scale.

   Note: For IBM Spectrum Scale 4.2.0.3 or 4.2.1 and later, the restart of IBM Spectrum Scale can be delayed until the next step, because tuning workerThreads will require a restart.

   Set each configuration option individually:

   ```
   mmchconfig readReplicaPolicy=local
   mmchconfig unmountOnDiskFail=meta
   mmchconfig maxStatCache=512 or mmchconfig maxStatCache=0
   mmchconfig restripeOnDiskFailure=yes
   mmchconfig nsdThreadsPerQueue=10
   mmchconfig nsdMinWorkerThreads=48
   mmchconfig prefetchaggressivenesswrite=0
   mmchconfig prefetchaggressivenessread=2
   ```

Note: The stat cache is not effective on the Linux platform unless the Local Read-Only Cache (LROC) is configured. Therefore on Linux, the setting for **maxStatCache** depends on whether the Local Read-Only Cache (LROC) is enabled on the node:

* If LROC is not enabled on the node, set **maxStatCache** to 0.
* If LROC is enabled on the node, set **maxStatCache** to an appropriate nonzero value based on the number of nodes in the cluster, the number of token managers in the cluster, the size of the LROC device, and any other relevant factors. For more information about LROC, see the topic *Local read-only cache* in the *IBM Spectrum Scale: Administration Guide*.

This restriction applies to all the versions and distributions of Linux that IBM Spectrum Scale supports.

Set all the configuration option at once by using the **mmchconfig** command:

```
mmchconfig readReplicaPolicy=local,unmountOnDiskFail=meta,maxStatCache=512,
restripeOnDiskFailure=yes,nsdThreadsPerQueue=10,nsdMinWorkerThreads=48,
prefetchaggressivenesswrite=0,prefetchaggressivenessread=2
```

The maxMBpS tuning option must be set as per the network bandwidth available to IBM Spectrum Scale. If you are using one 10 Gbps link for the IBM Spectrum Scale network traffic, the default value of 2048 is appropriate. Otherwise scale the value of maxMBpS to be about twice the value of the network bandwidth available on a per node basis.

For example, for two bonded 10 Gbps links an appropriate setting for maxMBpS is:

```
mmchconfig maxMBpS=4000 # this example assumes a network bandwidth of about
2GB/s (or 2 bonded 10 Gbps links) available to Spectrum Scale
```

Note: In a special user scenario (e.g. active-to-active disaster recovery deployment), `restripeOnDiskFailure` must be configured as no for internal disk cluster.

Some of these configuration options do not take effect until IBM Spectrum Scale is restarted.

10. Depending on the level of code installed, follow the tuning recommendation for Case A or Case B:

    a. If running IBM Spectrum Scale 4.2.0 PTF3, 4.2.1, or any higher level, either set workerThreads to 512, or try setting workerThreads=8*cores per node (both require a restart of IBM Spectrum Scale to take effect). For lower code levels, setting worker1Threads to 72 (with the **-i**, immediate, option to **mmchconfig** does not require restarting IBM Spectrum Scale.)

       ```
       mmchconfig workerThreads=512 # for Spectrum Scale 4.2.0 PTF3, 4.2.1, or
       any higher levels
       or
       mmchconfig workerThreads=8*CORES_PER_NODE # for Spectrum Scale 4.2.0 PTF3,
       4.2.1, or any higher levels
       ```

       Change workerThreads to 512 (the default is 128) to enable additional thread tuning. This change requires that IBM Spectrum Scale be restarted to take effect.

       Note: For IBM Spectrum Scale 4.2.0.3 or 4.2.1 or later, it is recommended that the following configuration parameters not be changed (setting workerThreads to 512, or (8*cores per node), will auto-tune these values): parallelWorkerThreads, logWrapThreads, logBufferCount, maxBackgroundDeletionThreads, maxBufferCleaners, maxFileCleaners, syncBackgroundThreads, syncWorkerThreads, sync1WorkerThreads, sync2WorkerThreads, maxInodeDeallocPrefetch, flushedDataTarget, flushedInodeTarget, maxAllocRegionsPerNode, maxGeneralThreads, worker3Threads, and prefetchThreads.

       After you enable auto-tuning by tuning the value of workerThreads, if you previously changed any of these settings (parallelWorkerThreads, logWrapThreads, etc) you must restore them back to their default values by running **mmchconfig <tunable>=Default**.

    b. For IBM Spectrum Scale 4.1.0.x, 4.1.1.x, 4.2.0.0, 4.2.0.1, 4.2.0.2, the default values will work for most scenarios. Generally only worker1Threads tuning is required:

       ```
       mmchconfig worker1Threads=72 -i # for Spectrum Scale 4.1.0.x, 4.1.1.x,
       4.2.0.0, 4.2.0.1, 4.2.0.2
       ```

For IBM Spectrum Scale 4.1.0.x, 4.1.1.x, 4.2.0.0, 4.2.0.1, 4.2.0.2, worker1Threads=72 is a good starting point (the default is 48), though larger values have been used in database environments and other configurations that have many disks present.

11. Customers running IBM Spectrum Scale 4.1.0, 4.1.1, and 4.2.0 must change the default configuration of trace to run in overwrite mode instead of blocking mode.

To avoid potential performance problems, customers running IBM Spectrum Scale 4.1.0, 4.1.1, and 4.2.0 must change the default IBM Spectrum Scale tracing mode from blocking mode to overwrite mode as follows:

```
/usr/lpp/mmfs/bin/mmtracectl --set --trace=def --tracedev-writemode=
overwrite --tracedev-overwrite-buffer-size=500M # only for Spectrum
Scale 4.1.0, 4.1.1, and 4.2.0
```

This assumes that 500MB can be made available on each node for IBM Spectrum Scale trace buffers. If 500MB are not available, then set a lower appropriately sized trace buffer.

12. Consider whether pipeline writing must be enabled.

By default, data ingestion node writes 2 or 3 replicas of the data to the target nodes over the network in parallel when pipeline writing is disabled (enableRepWriteStream=0). This takes additional network bandwidth. If pipeline writing is enabled, the data ingestion node only writes one replica over the network and the target node writes the additional replica. Enabling pipeline writing (**mmchconfig enableRepWriteStream=1** and restarting IBM Spectrum Scale daemon on all nodes) can increase IO write performance in the following two scenarios:

a. Data is ingested from the IBM Spectrum Scale client and the network bandwidth from the data-ingesting client is limited.

b. Data is written through rack-to-rack switch with limited bandwidth. For example, 30 nodes per rack, the bandwidth of rack-to-rack switch is 40Gb. When all the nodes are writing data over the rack-to-rack switch, each node will only get 40Gb/30, which is approximately 1.33Gb average network bandwidth.

For other scenarios, enableRepWriteStream must be kept as 0.

Optional IBM Spectrum Scale configuration and tuning

Note: A restart of IBM Spectrum Scale is needed to bring into effect any configuration option changes that do not successfully complete with the **-i** (immediate) option to **mmchconfig**. For example, changing the minMissedPingTimeout requires a restart.

The following configurations can be changed by using **mmchconfig** as per the needs of the system workload:

Configuration Options	Default	Recommended	Comment
metadataDiskWaitTimeForRecovery	2400(seconds)	Refer comment	Effective when restripeOnDiskFailure is set to yes. The default value is 40 minutes. This value must be long enough to cover the reboot time of the node with meta disk.
dataDiskWaitTimeForRecovery	3600(seconds	Refer comment	Effective when restripeOnDiskFailure is set to yes. The default value is 60 minutes. This value must be long enough to cover the reboot time of the node with data disk.

Configuration Options	Default	Recommended	Comment
syncBuffsPerIteration	100	100 (default)	It is recommended to not change the default value. Substantial improvements resulting from tuning this value have not been observed.
minMissedPingTimeout	3(seconds)	10-60(seconds)	Sets the lower bound on a missed ping timeout. For FPO clusters, a longer grace time is desirable before marking a node as dead, as it impacts all associated disks. Additionally, when running MapReduce workloads, the CPU can become overly busy and cause delayed ping responses. However, a longer timeout implies delay in recovery. A value between 10– 60 seconds is recommended. This value generally provides a good balance between the time to detect the real failures and the rate of false failure detection triggered by a delayed ping response due to CPU or network overload.
leaseRecoveryWait	35	65	The recommended value is 65 but the customers can choose lower values if they desire more rapid detection of failures, at the risk of experiencing temporary outages when nodes are heavily loaded or experience network problems.
prefetchPct	20(% of **pagepool** parameter)	See comment	Used by IBM Spectrum Scale as a guideline to limit the page pool space used for prefetch or write-behind buffers. For MapReduce workloads, generally used for sequential read and write, increase this parameter up to its 60% of the maximum **pagepool** size.

Configuration Options	Default	Recommended	Comment
maxFilesToCache	4000	100000	Specifies the number of inodes to cache. Storing the inode of a file in cache permits faster re-access to the file while retrieving location information for data blocks. Increasing this number can improve throughput for workloads with high file reuse, as is the case with Hadoop MapReduce tasks. However, increasing this number excessively can cause paging at the file system manager node. The value must be large enough to handle the number of concurrently open files and allow caching of recently used files.
nsdInlineWriteMax	1,024	1,000,000 (or default tuning value)	Defines the amount of data sent in-line with write requests to the NSD server and helps to reduce overhead caused by internode communication.
nsdThreadsPerDisk	3	8	For NSDs that are each SATA/SAS disk, set 8 for file systems with the block size of 2MB and 16 for file systems with the block size of 1MB. For NSDs that are LUNs, each containing multiple physical disks, increase this accordingly to the number of drives per NSD.
nsdSmallThreadRatio	0	2	The ratio of the number of small threads to the number of large threads. The recommendation is to change this to 2 for most workloads.

Configuration and tuning of Hadoop workloads

All configuration options listed in this section are applicable only to Hadoop-like applications such as Hadoop and Spark:

Configuration	Default Value	Recommended	Comment
forceLogWriteOnFdatasync	Yes	No	
disableInodeUpdateOnFdatasync	No	Yes	
dataDiskCacheProtectionMethod	0	2	Change this to 2 if you turn ondataOnly disk write cache (without battery protection).

Note: If the cluster is not dedicated for Hadoop workloads, take the default value for the above configurations.

For Hadoop-like workloads, one JVM process can open a lot of files. Therefore, tune the `ulimit` values:

```
vim /etc/security/limits.conf
# add the following lines at the end of /etc/security/limits.conf
* soft nofile 65536
* hard nofile 65536
* soft nproc 65536
* hard nproc 65536
```

kernel.pid_max

Usually, the default value is 32K. If you see the error `allocate memory` or `unable to create new native thread`, try to increase `kernel.pid_max` by adding `kernel.pid_max=99999` at the end of /etc/sysctl.conf and then `sysctl -p`.

Configuration and tuning of database workloads

If the cluster is focused on database workloads such as SAP HANA/DB2® DPF/DashDB or DIO/AIO, the following configuration options must be tuned:

Configuration	Default Value	Recommended	Comment
enableLinuxReplicatedAio	N/A	Yes	The default value depends on the release of IBM Spectrum Scale.
preStealPct	1	See below	Only for Direct I/O.

Database workload customers using direct I/O must also enable the following `preStealPct` tuning depending on the IBM Spectrum Scale levels:

- 3.5 (any PTF level)
- 4.1.1 (below PTF 10)
- 4.2.0 (any PTF level)
- 4.2.1 (below PTF 2).

The database workload customers with direct I/O enabled who are running older code levels must tune `preStealPct` as follows:

```
echo 999 | mmchconfig preStealPct=0 -i
```

After upgrading to IBM Spectrum Scale from one of the previously referenced older code levels to a higher level (especially 4.1.1 PTF 10, 4.2.1 PTF 2, or 4.2.2.0 or higher), you can set the configuration option `preStealPct=0` to its default value as follows:

```
echo 999 | mmchconfig preStealPct=1 -i
```

Configuring and tuning SparkWorkloads

1. Configure `spark.shuffle.file.buffer`.

   By default, this must be configured on $SPARK_HOME/conf/spark-defaults.conf.

   To optimize Spark workloads on an IBM Spectrum Scale file system, the key tuning value to set is the `spark.shuffle.file.buffer` configuration option used by Spark (defined in a spark config file) which must be set to match the block size of the IBM Spectrum Scale file system being used.

   The user can query the block size for an IBM Spectrum Scale file system by running: **mmlsfs <filesystem_name> -B**

   The following is an example of tuning the `spark_shuffle_buffer_size` for a given file system:

```
spark_shuffle_file_buffer=$(/usr/lpp/mmfs/bin/mmlsfs
<filesystem_name> -B | tail -1 | awk ' { print $2} ')
```

Need to set the Spark configuration option `spark.shuffle.file.buffer` to the value assigned to `$spark_shuffle_file_buffer`.

Defining a large block size for IBM Spectrum Scale file systems used for spark shuffle operations can improve system performance. However, using a block size larger than 2M can offer useful improvements on typical hardware used in FPO configurations is not proven.

2. Configure `spark.local.dir` with local path.

Do not put the Spark's shuffle data into the IBM Spectrum Scale file system because this slows down the shuffle process.

Ingesting data into IBM Spectrum Scale clusters

`MapReduce` tasks perform best when input data is evenly distributed across cluster nodes. You can use the following approaches or a combination to ingest data for the first time and on an ongoing basis:

- Import data through a diskless IBM Spectrum Scale node. This ensures that the data is distributed evenly across all failure groups and all nodes within a failure group.
- If you have a large set of data to copy, it might help to use all cluster nodes to share ingest workload. Use a **write-affinity** depth of 0, along with as many cluster nodes with storage as possible to copy data in parallel.
- A **write-affinity** depth of 0 ensures that each node distributes data across as many nodes as possible. IBM Spectrum Scale policies can be used to enforce write-affinity depth settings based on fileset name, filename, or other attributes.
- Another mechanism to distribute data on ingest is to use **write-affinity depth failure** groups (WADFG) to control placement of the file replica. A WADFG setting of "*,*,*" ensures that all the file chunks are evenly distributed across all nodes. A placement policy can be used to selectively specify this attribute on the data set being ingested.

It is possible that even after employing the above techniques to ingest, the cluster might become unbalanced as nodes and disks are added or removed. You can check whether the data in the cluster is balanced by using the **mmdf** command. If data disks in different nodes are showing uneven disk usage, rebalance the cluster by running the **mmrestripefs -b** command. Keep in mind that the rebalancing command causes additional I/O activity in the cluster. Therefore, plan to run it at a time when workload is light.

Exporting data out of IBM Spectrum Scale clusters

In many applications, it might be required to export the output data into another system or application for further use. Hadoop native components such as Flume can be used for this purpose. If HDFS Transparency is used for Hadoop applications, the distcp feature is supported over IBM Spectrum Scale to export data into a remote HDFS file system or IBM Spectrum Scale file system. Additionally, since IBM Spectrum Scale provides POSIX semantics, custom scripts can be written to move data from an IBM Spectrum Scale cluster to any other POSIX-compliant file system. Consider using CNFS to copy the needed data directly.

If data must be exported into another IBM Spectrum Scale cluster, the AFM function can be used to replicate data into a remote IBM Spectrum Scale cluster.

Upgrading FPO

When the application that runs over the cluster can be stopped, you can shut down the entire GPFS cluster and upgrade FPO. However, if the application over the cluster cannot be stopped, you need to take the rolling-upgrade procedure to upgrade nodes.

During this kind of upgrade, the service is interrupted. In production cluster, service interrupt is not accepted by the customers. If such cases, you need to take the rolling upgrade to upgrade node by node (or failure group by failure group) while keeping GPFS service up in the other nodes.

The guide for rolling upgrade is as follows:
- Only upgrade nodes from the same failure group at the same time slot; not operate nodes from two or more failure groups because bringing nodes from more than 1 failure groups will make your data exposed in data lost risk.
- Not break the quorum relationship when bringing down the nodes from one failure group. Before you bring down the nodes in one failure group, you need to check the quorum node. If bringing the quorum node in the to-be-operated failure group will break the quorum relationship in the cluster, you need to exclude that node for the rolling upgrade of the failure group.

Prerequisites
- Ensure that all disks are in a `ready` status and up availability. You can check by issuing the **mmlsdisk fs-name -L** command.
- Verify whether the upgraded-to GPFS version is compatible with the running version from IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ gpfsclustersfaq.html). For example, you cannot upgrade GPFS from 3.4.0.x directly into 3.5.0.24. You need to upgrade to 3.5.0.0 first and then upgrade to the latest PTF. You also need to verify whether the operating system kernel version and the Linux distro version are compatible with GPFS from IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ gpfsclustersfaq.html).
- Find a time period when the whole system work load is low or reserve a maintenance time window to do the upgrade. When cluster manager or file system manager is down intentionally or accidentally, another node is elected to take the management role. But it takes time to keep the cluster configuration and the file system data consistent.
- When a file system manager is elected by cluster manager, it does not change even if the file system is unmounted in this node. If the file system is mounted in other nodes, it is also 'internal' mounted in the file system manager. This does not affect your ability to unload the kernel modules and upgrade GPFS without a reboot.

Upgrade FPO as follows:
1. Disable auto recovery for disk failure

   To do a rolling upgrade of GPFS, you must shut down GPFS in some nodes. Disks in those nodes are unreachable during that time. It is better to handle this disk down manually with the following step.

   Run **mmchconfig restripeOnDiskFailure=no -i** in any node in cluster to disable auto recovery for disk failure. It is necessary to include **-i** option for this change to take effect immediately and permanently. GPFS **restripeOnDiskFailure** is a cluster-wide configuration. So you need to run it only once in any one node in your cluster.
2. Get the list of nodes for this upgrade cycle

   Each upgrade cycle, you can only upgrade GPFS in nodes whose disks in it have the same first two numbers in failure group vector. Save node list in file **nodeList**, one node name per line in it. For general information on how to specify node names, see "Specifying nodes as input to GPFS commands" on page 103.
3. Get disk list in nodes for this upgrade cycle

   Get all disks attached in nodes for this upgrade cycle. Save it in file **diskList**. Each line in file **diskList** saves a disk name. **mmlsdisk —L** command can show which disks belong to the nodes you want to upgrade in this cycle.
4. Stop applications by using GPFS file system

Confirming that there is no application which still opens file in GPFS file system. You can use **lsof** or **fuse** command to check whether there is still an open instance active for file in GPFS file system.

5. Unmount GPFS file system

   Unmount GPFS file system in all nodes for this upgrade cycle through command:

   ```
   mmumount <fsName> -N <nodeList>
   ```

   Confirming file system has already unmounted in all related nodes through command:

   ```
   mmlsmount <fsName> -L
   ```

6. Suspend disks

   Suspend all attached disks in nodes for this upgrade cycle to make sure that GPFS does not try to allocate new data block from these disks. GPFS can still read valid data block from suspended disk

   ```
   mmchdisk <fsName> suspend -d <diskList>
   ```

   Confirming disks are suspended properly through command:

   ```
   mmlsdisk <fsName>
   ```

7. Shut down GPFS

   Shut down GPFS in all nodes for this upgrade cycle through command:

   ```
   mmshutdown -N <nodeList>
   ```

   Confirming GPFS is in `down` status in these nodes through command:

   ```
   mmgetstate -a
   ```

8. Upgrade GPFS

   You can upgrade GPFS packages in each node of this upgrade cycle.

   For general information on how to install GPFS packages on nodes, see the following topics in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*:

   - *Installing IBM Spectrum Scale on Linux nodes and deploying protocols*
   - *Installing IBM Spectrum Scale on AIX nodes*
   - *Installing IBM Spectrum Scale on Windows nodes*

9. Start GPFS

   After all packages are ready, you can start up GPFS:

   ```
   mmstartup -N <nodeList>
   ```

   Confirming GPFS are in "Active" status in these nodes through command:

   ```
   mmgetstate -a
   ```

10. Resume and start disks

    When GPFS is in "Active" status in all nodes, you can resume disks which were suspended intentionally in Step 7.

    ```
    mmchdisk <fsName> resume -a or
    mmchdisk <fsName> resume -d <diskList>
    ```

    When these disks are in "ready" status and if some of these disks are in "down" availability, you can start these disks through the following command:

    ```
    mmchdisk <fsName> start -a or
    mmchdisk <fsName> start -d <diskList>
    ```

    This might take a while since GPFS must do incremental data sync up to keep all data in these suspended disks are up to date. The time it needs depends on how much data has changed when the disks were kept in `suspended` status. You have to wait for **mmchdisk start** command to finish to do next step.

    Confirming all disks are in `ready` status and up state through command:

    ```
    mmlsdisk <fsName>
    ```

11. Mounts GPFS file system

    When all disks in the file system are in `up` status you can mount file system:

```
mmmount <fsName> -N <nodeList>
```

Confirming GPFS file system is mounted properly through command:

```
mmlsmount <fsName> -L
```

Repeat Step 3 to Step 12 to upgrade GPFS in all nodes in your cluster.

12. Enable auto recovery for disk failure

    Now you can enable auto recovery for disk failure through command:

    ```
    mmchconfig restripeOnDiskFailure=no -i
    ```

    It's necessary to includes **-i** option for this change to take effect immediately and permanently.

13. Upgrade GPFS cluster version and file system version

    Issue `mmchconfig release=LATEST` and `mmchfs -V compat` to ensure the upgrade is successful and the cluster would never revert to the old build for minor GPFS upgrade. It is recommended to use `mmchfs -V compat` to enable backward-compatible format changes.

    For major GPFS upgrade, consult IBM Support Center to verify compatibility between the different GPFS major versions, before issuing `mmchfs -V full`. For information about specific file system format and function changes when you upgrade to the current release, see Chapter 14, "File system format changes between versions of IBM Spectrum Scale," on page 153.

Monitoring and administering IBM Spectrum Scale FPO clusters

IBM Spectrum Scale supports the use of the simple network management protocol (SNMP) for monitoring the status and configuration of the IBM Spectrum Scale cluster. By using an SNMP application, the system administrator can get a detailed view of the system and receive instant notification of important events, such as a node or disk failure.

The SNMP agent software consists of a master agent and a set of subagents, which communicate with the master agent through an agent/subagent protocol, the AgentX protocol in this case.

The SNMP subagent runs on a collector node of the IBM Spectrum Scale cluster. The collector node is designated by the system administrator by using the `mmchnode` command.

The Net-SNMP master agent, also called as the SNMP daemon, or `snmpd`, must be installed on the collector node to communicate with the IBM Spectrum Scale subagent and with your SNMP management application. Net-SNMP is included in most Linux distributions and must be supported by your Linux vendor.

For more information about enabling SNMP support, see the *GPFS SNMP support* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

Refer to the GPFS SNMP support topic in the *IBM Spectrum Scale: Administration Guide* for further information about enabling SNMP support.

If using IBM BigInsights

When you install IBM Spectrum Scale, you can enable IBM Spectrum Scale monitoring using the IBM BigInsights installation program. If the monitoring was not enabled at the time of installation, it can be done later by installing the Net-SNMP master agent on the collector node to communicate with the IBM Spectrum Scale subagent and the IBM BigInsights Console. Detailed instructions are provided in the Enabling monitoring for GPFS topic in the IBM InfoSphere® BigInsights Version 2.1.2 documentation.

If using Platform Symphony® or Hadoop distribution from other vendor

You can leverage IBM Spectrum Scale SNMP integration for centralized monitoring of the IBM Spectrum Scale cluster. Follow the procedure outlined in the *GPFS SNMP support* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

Rolling upgrades

During a regular upgrade, the IBM Spectrum Scale service is interrupted. For a regular upgrade, you must shut down the cluster and suspend the application workload of the cluster. During a rolling upgrade, there is no interruption in the IBM Spectrum Scale service. In a rolling upgrade, the system is upgraded node by node or failure group by failure group. During the upgrade, IBM Spectrum Scale runs on a subset of nodes.

In a rolling upgrade, nodes from the same failure group must be upgraded at the same time. If nodes from two or more failure groups stop functioning, only a single data copy is available online. Also, if the quorum node stops functioning, the quorum relationship in the cluster is broken. Therefore, the quorum node must be excluded from the rolling upgrade of the failure node.

Performing a rolling upgrade

This topic lists the steps to perform a rolling upgrade.

- Ensure that the status of all disks is Ready and the availability is Up by running the `mmlsdisk <fs-name> -L` command.
- Verify the compatibility of the new IBM Spectrum Scale version with the running version by reviewing the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html). For example, IBM Spectrum Scale cannot be upgraded from 3.4.0.x to 3.5.0.24 before being upgraded to 3.5.0.0.
- Verify the compatibility of the planned upgrade system kernel and Linux distro versions with IBM Spectrum Scale by reviewing the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
- While performing maintenance on the cluster manager and the file system manager nodes, the nodes fail over automatically. However, you must manually assign the cluster manager and the file system manager to other nodes by using the `mmchmgr` command when the cluster is not busy.

1. Disable auto recovery for disk failure.

   To upgrade a node, shut down IBM Spectrum Scale running on the node. When IBM Spectrum Scale is shut down, disks in the node cannot be reached. Instead of letting the disks fail and the automatic recovery initiate, temporarily disable auto recovery.

   Run the `mmchconfig restripeOnDiskFailure=no -i` command to disable auto recovery for disk failure. With the `-i` option, the parameter takes effect immediately and permanently. For example, in small clusters, the node number is less than 30 nodes. Therefore, it takes a shorter time for IBM Spectrum Scale to synchronize the configuration. For large clusters, the node number is in hundreds. Therefore, the time taken to synchronize the configuration is longer. The `restripeOnDiskFailure` parameter is a cluster-wide configuration.

   After disabling auto recovery, check for auto recovery in the file system manager by running the following commands:

   - If there are multiple file systems in the cluster, run `mmlsmgr` command to check the fs manager of a single file system.
   - Log in to the fs manager of the file system and run `ps -elf | grep -e tschdisk -e tsrestripefs` command. If there are processes running, wait for them to complete.

2. Select the nodes that must be upgraded and schedule the time of each upgrade. In each upgrade cycle, you can only upgrade IBM Spectrum Scale on nodes where the disks have the same first two numbers in the failure group. Save the list of nodes in the nodeList file with one node name on each line. Save a list of the disks on the nodes that will be upgraded in this cycle in the diskList file, with each line containing an NSD name. Run the `mmlsdisk Device –M` command to check which disks belongs to which node.

3. Stop all applications that are using the IBM Spectrum Scale file system before stopping IBM Spectrum Scale. To check for open files in the file system, run the `lsof` or the `fuse` command.

4. Unmount the IBM Spectrum Scale file system on all nodes by running the following command: `mmumount <fsName> -N <nodeList>`

   To confirm that the file system has been unmounted on all related nodes, run the following command: `mmlsmount <fsName> -L`

5. Suspend all disks in the nodes so that IBM Spectrum Scale does not allocate new data blocks from these disks. IBM Spectrum Scale can still read data block from suspended disks by running the following command:

   `mmchdisk <fsName> suspend -d <diskList>`

   To confirm that all disks are suspended properly, run the following command: `mmlsdisk <fsName>`

6. Shut down IBM Spectrum Scale on the nodes by running the following command:

   `mmshutdown -N <nodeList>`

   To confirm IBM Spectrum Scale has stopped functioning on these nodes, run the following command: `mmgetstate -a`

   Upgrade IBM Spectrum Scale packages on each node. For information on how to install IBM Spectrum Scale packages on node, see the following topics:

   - *Installing IBM Spectrum Scale on Linux nodes and deploying protocols* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
   - *Installing IBM Spectrum Scale on AIX nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
   - *Installing IBM Spectrum Scale on Windows nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

   After everything has been installed and the portability layer has been built, start IBM Spectrum Scale by running the following command: `mmstartup -N <nodeList>`

   To confirm that IBM Spectrum Scale is active on the upgraded nodes, run the following command: `mmgetstate -a`.

   Resume all the suspended disks by running the following commands: `mmchdisk <fsName> resume -a` or `mmchdisk <fsName> resume -d <diskList>`.

   If some of the suspended disks are in the Down availability, start these disks by running the following command: `mmchdisk <fsName> start -a` or `mmchdisk <fsName> start -d <diskList>`.

   This may take a while because IBM Spectrum Scale is performing an incremental data sync up to keep the data in these suspended disks up-to-date. The time taken depends on the data that has been changed while the disks were kept in the Suspended status. Wait for the `mmchdisk <fsName> start [. . .]` command to finish before moving on to the next step.

   To confirm that all disks are in the ready state, run the following command: `mmlsdisk <fsName>`.

7. When all the disks in the file system are functioning, mount the file system by running the following command: `mmmount <fsName> -N <nodeList>`

   Confirm that the IBM Spectrum Scale file system has mounted by running the following command: `mmlsmount <fsName> -L`

8. Perform Step through Step to upgrade IBM Spectrum Scale on all nodes in the cluster.

9. To enable auto recovery for disk failure, run the following command: `mmchconfig restripeOnDiskFailure=yes -i`

   Ensure that you use the `-i` option so that this change takes effect immediately and permanently.

10. Upgrade the IBM Spectrum Scale cluster version and file system version

    If all applications run without any issues, run the `mmchconfig release=LATEST` command to upgrade the cluster version to the latest. Then, run the `mmchfs -V compat` command to ensure that the upgrade is successful. To enable backward-compatible format changes, run `mmchfs -V compat`.

Note: After running the `mmchconfig release=LATEST` command, you cannot revert the cluster release version to an older version. After running the `mmchfs -V compat` command, you cannot revert the file system version to an older version.

For major IBM Spectrum Scale upgrade, check IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) or contact `scale@us.ibm.com` before running the `mmchfs -V full` command to verify the compatibility between the different IBM Spectrum Scale major versions. For information about specific file system format and function changes, see Chapter 14, "File system format changes between versions of IBM Spectrum Scale," on page 153.

Upgrading other infrastructure

The same process of choosing nodes should be used when upgrading hardware firmware, operation system kernel or other components that require you to take IBM Spectrum Scale down on the node.

The IBM Spectrum Scale FPO cluster

When a node reboots due to hardware or software issues, IBM Spectrum Scale can be started and the file system can be mounted if autoload is configured as yes in the mmchconfig command. In a typical FPO deployment, each node has several local attached disks. When a node stops functioning, disks attached to the node are made unavailable from the cluster.

Note: Do not reboot a node if the file system is still mounted.

After the node is rebooted, the disk status of the node is uncertain. The status of the node is dependent upon the auto recovery configuration (`mmlsconfig restripeOnDiskFailure`) and the IO operations over the cluster.

Restarting a large IBM Spectrum Scale cluster

A cluster might have to be restarted because of an OS upgrade. On large FPO clusters auto-recovery must be disabled before restarting IBM Spectrum Scale.

* Ensure that the status of all disks is Ready and the availability is Up by running the `mmlsdisk <fs-name> -L` command.
* Verify the compatibility of the planned upgrade system kernel and Linux distro versions with IBM Spectrum Scale by reviewing the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

1. Disable auto recovery for disk failure.

   When IBM Spectrum Scale stops functioning, some nodes might not shut down. This might bring some disks down from the fast nodes and might trigger auto recovery. To avoid this, temporarily disable auto recovery.

   Run the `mmchconfig restripeOnDiskFailure=no -i` command to disable auto recovery for disk failure. With the `-i` option, the parameter takes effect immediately and permanently. For example, in small clusters, the node number is less than 30 nodes. Therefore, it takes a shorter time for IBM Spectrum Scale to synchronize the configuration. For large clusters, the node number is in hundreds. Therefore, the time taken to synchronize the configuration is longer. The `restripeOnDiskFailure` parameter is a cluster-wide configuration.

   After disabling auto recovery, check for auto recovery in the file system manager by running the following commands:

   * If there are multiple file systems in the cluster, run `mmlsmgr` command to check the fs manager of a single file system.
   * Log in to the fs manager of the file system and run `ps -elf | grep -e tschdisk -e tsrestripefs` command. If there are processes running, wait for them to complete.

2. Stop all applications that are using the IBM Spectrum Scale file system. To check for open files in the file system, run the `lsof` or the `fuse` command.

For example, to check if IBM Spectrum Scale file system has processes using it run the following commands: `lsof +f -- /dev/name_of_SpectrumScale_filesystem` or `fuser -m /mount_point_of_SpectrumScale_filesystem`

3. Unmount the IBM Spectrum Scale file system on all nodes for this upgrade cycle by running the following command: `mmumount <fsName> -a`

   To confirm that the file system has been unmounted on all related nodes, run the following command: `mmlsmount <fsName> -L`

4. To disable the Automatic mount option, run the following command: `mmchfs <fsName> -A no`

   Note: In a large cluster, some nodes might take a while to start. If the `-A` option is not set to **no**, unnecessary disk IO might cause some disks from slow nodes to be marked as non functional.

5. Shut down IBM Spectrum Scale on the nodes by running the following command:

   `mmshutdown -N <nodeList>`

   To confirm IBM Spectrum Scale has stopped functioning on these nodes, run the following command: `mmgetstate -a`

6. Upgrade IBM Spectrum Scale or perform the maintenance procedure on the whole cluster.

7. Start IBM Spectrum Scale cluster. After everything has been installed and the portability layer has been built, start IBM Spectrum Scale by running the following command: `mmgetstate -a`.

   To confirm that IBM Spectrum Scale is active on the upgraded nodes, run the following command: `mmgetstate -a`.

8. When IBM Spectrum Scale is active on all nodes, check the state of all disks by running the following command: `mmlsdisk <fsName> -e`. If some disks in the file system do not have the Up availability and the Ready status, run the `mmchdisk <fsName> start -a` command so that the disks start functioning. Run the `mmchdisk <fsName> resume -a` command so that the suspended and to-be-emptied disks become available.

9. When all the disks in the file system are functioning, mount the file system by running the following command: `mmmount <fsName> -N <nodeList>`

   Confirm that the IBM Spectrum Scale file system has mounted by running the following command: `mmlsmount <fsName> -L`

10. To enable auto recovery for disk failure, run the following command: `mmchconfig restripeOnDiskFailure=yes -i`

    Ensure that you use the `-i` option so that this change takes effect immediately and permanently.

11. To enable the Automatic mount option, run the following command: `mmchfs <fsName> -A yes`.

12. If you have upgraded IBM Spectrum Scale version in step 6, upgrade the IBM Spectrum Scale cluster version and file system version.

    If all applications run without any issues, run the `mmchconfig release=LATEST` command to upgrade the cluster version to the latest. Then, run the `mmchfs -V compat` command to ensure that the upgrade is successful. To enable backward-compatible format changes, run `mmchfs -V compat`.

    Note: After running the `mmchconfig release=LATEST` command, you cannot revert the cluster release version to an older version. After running the `mmchfs -V compat` command, you cannot revert the file system version to an older version.

    For major IBM Spectrum Scale upgrade, check IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) or contact `scale@us.ibm.com` before running the `mmchfs -V full` command to verify the compatibility between the different IBM Spectrum Scale major versions. For information about specific file system format and function changes, see Chapter 14, "File system format changes between versions of IBM Spectrum Scale," on page 153.

Failure detection

The node state
Learn how to find the state of the nodes in an IBM Spectrum Scale cluster.

To check the node state, issue the **mmgetstate** command with or without the **-a** option, as in the following
examples:
```
1) mmgetstate
```
```
2) mmgetstate -a
```

Be aware of the differences between the **down**, **unknown**, and **unresponsive** states:
• A node in the **down** state is reachable but the GPFS daemon on the node is not running or is recovering
from an internal error.
• A node in the **unknown** state cannot be reached from the node on which the **mmgetstate** command was
run.
• A node in the **unresponsive** state is reachable but the GPFS daemon on the node is not responding.

To follow up on investigating the state of a node, check if the node is functioning or has a network issue.

For more information, see the topic *mmgetstate command* in the *IBM Spectrum Scale: Command and*
Programming Reference.

The disk state
This topic describes how to check the disk state in a IBM Spectrum Scale cluster.

To check the state of the disks in a IBM Spectrum Scale cluster, run **mmlsdisk -e** command. This
command lists all the disks that do not have the Available or Up status.

IBM Spectrum Scale log
This topic describes the IBM Spectrum Scale log files.

The IBM Spectrum Scale log files are saved in the `/var/adm/ras/` directory on each node. Each time the
IBM Spectrum Scale daemon starts, a new log file is created. The mmfs.log.latest log file is the link to the
latest log. On Linux, all additional information is sent to the system log in `/var/log/messages`.

Because the IBM Spectrum Scale cluster manager and file system managers handle cluster issues such as
node leaves or disk down events, monitor the IBM Spectrum Scale log on the cluster manager and file
system manager to get the best view of the cluster and file system status.

Disk Failures
This section describes how to handle a disk failure.

In an FPO deployment model with IBM Spectrum Scale the **restripeOnDiskFailure=yes** configuration
parameter should be set to yes. When a disk is not functioning, auto recovery enables the disk to start
functioning. Auto recovery enlists the help of any node in the cluster to help recover data. This may
affect the file system I/O performance on all nodes, because data might have to be copied from a valid
disk to recover the disk.

Stopping the disk failure auto recovery operation
The auto recovery operation can impact the IO performance across the cluster. Therefore, you can stop
auto recovery manually and restart it when the cluster is not so busy. The disks that are not functioning
must be recovered to protect your data.

Run the **mmlsdisk -e** command to see the disks that do not have the Up availability and the Ready status. If all the disks in the file system are functioning correctly, the system displays the following message: 6027-623 All disks up and ready.

1. To stop the auto recovery process, stop the **tschdisk** and **tsrestripefs** processes on the file system manager node. Log in to the IBM Spectrum Scale file system manager node. Retrieve the **tschdisk** and **tsrestripefs** command processor ID through the **ps -elf | grep -e tschdisk -e tsrestripefs** command.

   Alternatively, check the IBM Spectrum Scale log (/var/adm/ras/mmfs.log.latest) in the file system manager node to see if there is **tschdisk** command is still running. When the **restripefs** command is invoked by the auto recovery and is still running, the command log message is redirected to /var/adm/ras/restripefsOnDiskFailure.log.<timestamp>(IBM Spectrum Scale 4.1 and IBM Spectrum Scale 4.1.1) or /var/adm/ras/autorecovery.log.<timestamp>(IBM Spectrum Scale 4.1.1 PTF1 and later).

2. Stop the **tschdisk** and **tsrestripefs** command processes by running the **kill -9 <pid>** command. Double check whether **tschdisk** and **tsrestripefs** command processes have been stopped by running the **ps -elf | grep -e tschdisk -e tsrestripefs** command.

Starting the disk failure recovery

This topic lists the steps to start the disk failure recovery.

1. To check the disks that are not in the Ready state, run the **mmlsdisk -e** command.

2. Resume the disks that have the Suspended status.

   If there are multiple suspended disks, create a file that lists all the suspended disks one nsd name per line before you resume the disks. Resume the suspended disks by issuing the following command:

   mmchdisk <fsName> resume -d <suspendDisk:List>

   Check the disk status again by running the **mmlsdisk -e** command and confirm that all disks are now in the Ready state. If some disks are still in the Suspended state, there might be a hardware media or a connection problem. Save the names of these disks in the brokenDiskList file.

3. Save the disks that do not have the Up availability in the downDiskList file. Each line in downDiskList file stores a disk name. Start these disks by running the following command:

   mmchdisk <fsName> start -d <downDiskList>

   Check the disk status by running the **mmlsdisk -e** command to confirm that all disks have the Up availability. Disks that do not have the Up availability might have a hardware media or connection problem. Save the names of these disks in the tobeSuspendDiskList file. Suspend these disks by running the following command:

   mmchdisk <fsName> suspend -d <tobeSuspendDiskList>

4. If a disk is in Suspended status after you restart it, there might be a hardware media or connection problems. To keep your data safe, migrate it to the suspended disks by running the following command:

   mmrestripefs <fsName> -r

   After a file system restripe, all data in the suspended disks is migrated to other disks. At this point, all the data in the file system has the desired level of protection.

5. Check the disk connections and the disk media for disks that are in the Suspended state and repeat step 2 through step 4. If a failure occurs again, delete these disks from file system by running the **mmdeldisk** command. For example,

   mmdeldisk <fs-name> "broken-disk1;broken-disk2"

   If you are unable to delete a broken disk, contact IBM support.

Handling physically broken disk

If a disk is physically broken, it cannot be recovered by auto recovery or manual recovery. Do not keep broken disks in the file system and schedule to delete them from the file system.

Deleting disks when auto recovery is not enabled (check this by `mmlsconfig` `restripeOnDiskFailure`):

Deleting NSD disks from the file system can trigger disk or network traffic because of data protection. If your cluster is busy with application IO and the application IO performance is important, schedule a maintenance window to delete these broken disks from your file system. Follow the steps in the "Starting the disk failure recovery" on page 538 section to check if a disk is physically broken and handle the broken disks.

Deleting disks when auto recovery is enabled (check this by `mmlsconfig` `restripeOnDiskFailure`):

When the IO operation is being performed on the physically broken disks, IBM Spectrum Scale marks the disks as non functional. Auto recovery suspends the disks if it fails to change the availability of the disk to Up and restripes the data off the suspended disks. If you are using IBM Spectrum Scale 4.1.0.4 or earlier, deleting the non functional disks triggers heavy IO traffic (especially for metadata disks). On IBM Spectrum Scale 4.1.0.4, `mmdeldisk` command has been improved. If the data on non functional disks have been restriped, the disk status will be Emptied. The `mmdeldisk` command deletes the non functional disks with the Emptied status without involving additional IO traffic.

Node failure

In an FPO deployment, each node has locally attached disks. When a node fails or has a connection problem with other nodes in a cluster, disks in this node become unavailable. Reboot a node to repair a hardware issue or patch the operating system kernel. Both these cases are node failures.

If auto recovery is enabled, that is, the `restripeOnDiskFailure=yes` parameter is set to yes, and a failed node is recovered within auto recovery wait time (check the details described in the Auto Recovery for Disk Failure section), auto recovery handles the node failure automatically by bringing up down disks and ensuring all data has the desired replication. If a node is not recovered within the auto recovery wait time, auto recovery migrates the data off the disks in the failed node to other disks in cluster.

Reboot node intentionally

Automatic recovery of a node

If you want to reboot a node or enable some configuration change that requires a reboot and have it recovered without auto recovery, check the auto recovery wait time. The auto recovery wait time is defined by the minimum value of minDiskWaitTimeForRecovery, metadataDiskWaitTimeForRecovery and dataDiskWaitTimeForRecovery. By default, minDiskWaitTimeForRecovery is 1800 seconds, metadataDiskWaitTimeForRecovery is 2400 seconds and dataDiskWaitTimeForRecovery is 3600 seconds. If the reboot is completed within the auto recovery wait time, it is safe to unmount the file system, shut down IBM Spectrum Scale, and reboot your node without having to disable auto recovery.

Manual recovery of a node

When you want to perform hardware maintenance for a node that must be shut down for a long time, follow the same steps mentioned in IBM Spectrum Scale Rolling Upgrade Procedure and perform hardware maintenance.

Node crash and boot up

This topic lists the steps to recover a node automatically or manually.

Recovering a node automatically:
When a node crashes due to kernel or other critical issues and is recovered within the auto recovery wait period, IBM Spectrum Scale cluster manager can add this node automatically and IBM Spectrum Scale

auto recovery brings up disks and repairs the dirty data in disks attached in the node. Check if the node and the disks in the node work normally and the data in the disks is updated.

1. Check whether IBM Spectrum Scale state is active on all nodes in the cluster by running the `mmgetstate –a` command. If a node is functional but IBM Spectrum Scale is not active, check IBM Spectrum Scale log (`/var/adm/ras/mmfs.log.latest`) on the node and run the `mmstartup` command after the issue in the log has been resolved.

2. Check if any disk does not have the Up availability and the Ready state by running the `mmlsdisk -e` command. If all disks in the file system are in the Ready state, the system displays the following message `6027-623 All disks up and ready`. Perform the disk recovery operation to change the state of all disks to Ready.

3. Run the `mmlsdisk` command to confirm that there is no warning message at the end of output.

   If you see the following message, there are data replicas on suspended and to-be-emptied disks. If the suspended and to-be-emptied disks are not physically broken, recover them and run the `mmrestripefs -r` command to fix the warning message. If the disks are physically broken, suspend them and run the `mmrestripefs –r` command to fix the warning message.

   Attention: Due to an earlier configuration change the file system may contain data that is at risk of being lost.

Recovering a node manually:
When a node crashes and is not recovered while auto recovery is temporarily suspended, start the node and recover the disks. In this case, IBM Spectrum Scale auto recovery migrates all data from disks in this node to other valid disks in cluster.

1. Find the root cause of the node crash, fix it, and recover the node.

2. If IBM Spectrum Scale autoload configuration is disabled, start the IBM Spectrum Scale daemon by running the `mmstartup` command. Check if the node state is Active. If the state is not active after a few minutes, check IBM Spectrum Scale log (`/var/adm/ras/mmfs.log.latest`) on this node and run the `mmstartup` command after fixing the issue.

3. When IBM Spectrum Scale is active, auto recovery is invoked automatically to recover the disks in this node. Check the IBM Spectrum Scale log (`/var/adm/ras/mmfs.log.latest`) and the restripefs log (`/var/adm/ras/restripefsOnDiskFailure.log.latest`) on the file system manager node for more details.

Handling node crashes

If a failed node cannot be recovered, auto recovery migrates all data from disks in this node to other disks in cluster. If the system does not recover, delete the disks in the node and node.

1. Log in to another cluster node and run `mmlsdisk <fs-name> -M` command to get a list of disks attached to the failed node. Save the disk list in the diskList file. Each line lists a disk name.

2. Run the `mmdeldisk <fsName> -F <diskList>` command to delete the disks attached to the failed node.

3. Run the `mmdelnsd -F <diskList>` command to delete NSDs attached to the failed node. Run `mmdelnode` command to remove the node, or if you are replacing the node with new hardware, use the same name and IP address to continue.

   To replace the failed node with a new node, start the replacement mode with the hostname and the IP address of the failed node. Install IBM Spectrum Scale packages and configure SSH authorization with other nodes in the cluster. Run the following command to restore IBM Spectrum Scale configurations in this replacement node:

   `mmsdrrestore -p <cluster manager>  -R <remoteFileCopyCommand> -N <replacement node>`

   Use the `mmlsmgr` command to identify the cluster manager node. Use the Remote file copy command that is configured for the cluster.

4. Start IBM Spectrum Scale on the replacement node by running the `mmstartup -N <replacement node>` command. Confirm that IBM Spectrum Scale state is active by running the `mmgetstate –N <replacement node>` command.

5. Prepare a stanza file to create NSDs by running the **mmcrnsd** command and add these disks into file system by running the **mmadddisk** command.

Handling multiple nodes failure

Auto recovery must be enabled in an FPO cluster to protect data from multiple node failures. Set **mmchconfig restripeOnDiskFailure=yes -N all**.

Usually, if the concurrent failed nodes are less than maxFailedNodesForRecovery, auto recovery will protect data against node failure or disk failure. If the concurrent failed nodes are larger than maxFailedNodesForRecovery, auto recovery exits without any action and the administrator has to take some actions to recover it.

Multiple nodes failure without SGPanic

This topic lists the steps to handle multiple nodes failure without SGPanic

1. Recover the failed nodes.
2. If all nodes are recovered quickly, run the **mmlsdisk <fs-name> -e** command to view the down disk list.
3. Run the **mmlsnsd -X** command to check whether there are disks that are undetected by the operating system of nodes. For example,

```
# mmlsnsd -X

 Disk name    NSD volume ID      Device    Devtype  Node name                      Remarks
---------------------------------------------------------------------------------------------------------
 mucxs131d01  AC170E46561E7A8F   /dev/sdb  generic  mucxs131.muc.infineon.com server node
 mucxs131d02  AC170E46561E7A90   /dev/sdc  generic  mucxs131.muc.infineon.com server node
 mucxs531d07  AC170E4B5612838E   /dev/sdh  generic  mucxs531.muc.infineon.com server node
 mucxs531d08  AC170E4B56128391   -         -        mucxs531.muc.infineon.com (not found) server node
```

In the above output means the physical disk for the nsd mucxs531d08 is not recognized by the OS. If a disk is not detected, check the corresponding node to see if the disk is physically broken. If the undetected disks cannot be recovered quickly, remove them from the down disk list.

4. Run the **mmchdisk <fs-name> start -d <down disk in step3>**. If it succeeds, go to step5); if not, open PMR against the issue.
5. If the undetected disks cannot be recovered, run the **mmrestripefs <fs-name> -r** to fix the replica of the data whose part of replica are located in these undetected disks.

SGPanic for handling node failure

For internal disk rick storage (FPO clusters), unmountOnDiskFail must be configured as "meta". If it is not, change the configuration by running the **mmchconfig** command.

With unmountOnDiskFail configured as meta, if you see file system SGPanic reported when nodes are non functional, there are more than three nodes with metadata disk down together or there are more than three disks with meta data down. Follow the steps in the section 8.1 to fix the issues. Run the **mmfsck -n** command to scan the file system to ensure that mmfsck displays the following message: `File system is clean finally`. If mmfsck -n does not report "File system is clean", you need to open PMR to report the issues and fix this with guide from IBM Spectrum Scale.

Network switch failure

This topic describes how to handle network switch failure.

In an FPO cluster, if Auto recovery is enabled and there are more than maxFailedNodesForRecovery non functional nodes, auto recovery does not recover the nodes. By default, maxFailedNodesForRecovery is three nodes. You can change this number depending on your cluster configuration.

A switch network failure can cause nodes to be reported as non functional. If you want auto recovery to protect against switch network failures, careful planning is required in setting up the FPO cluster. For

example, a network switch failure must not bring disks (with metadata) down from 3 or more failure groups, and maxFailedNodesForRecovery must be configured to a value that is larger than the number of down nodes that will result from a switch network failure.

Data locality

In an FPO cluster, if the data storage pool is enabled with **allowWriteAffinity=yes**, the data locality is decided by the following order:
*   WADFG is set by **mmchattr** or the policy.
*   Default WAD or WAD is set by policy and the data ingesting node.

If the file is set with WADFG, the locality complies with WADFG independent of where the data is ingested. If the file is not set with WADFG, the locality is decided according to the WAD and data-ingesting node. Also, data locality configurations are the required configurations. If there are no disks available to comply with the configured data locality, the IBM Spectrum Scale FPO stores the data in other disks.

The data locality might be broken if there are **mmrestripefs -r** and **mmrestripefile** after disk failure or node failure. If your applications need data locality for good performance, restore the data locality after node failure or disk failure.

Data locality impacted from down disks

All disks in a node must be configured as the same failure or locality group. After a disk is nonfunctional, **mmrestripefs -r** from auto recovery suspends the disk and restripes the data on the nonfunctional disks onto other disks in the same locality group. The data locality is not broken because the data from local disks is still in that node. If you do not have other disks available in the same locality group, **mmrestripefs –r** from auto recovery restripes the data on the nonfunctional disks onto other nodes, breaking the data locality for the applications running over that node.

Data locality impacted from the nonfunctional nodes

If a nonfunctional node does not have NSD disks in the file system, the data locality is not impacted. If the nonfunctional node has NSD disks in the file system and the node is not recovered within dataDiskWaitTimeForRecovery (if all down disks are dataOnly disks) and metadataDiskWaitTimeForRecovery (if there is meta data NSD disk down), auto recovery suspends the disks and performs **mmrestripefs -r**. All disks from the nonfunctional nodes are not available for write and the data from the nonfunctional disks is restriped onto other nodes. Therefore, the data locality is broken on the nonfunctional nodes.

Data locality impacted from unintended **mmrestripefile -b** or **mmrestripefs –b**

If the file is not set with WADFG (by policy or by mmchattr), both **mmrestripefile -b** and **mmrestripefs -b** might break the data locality.

Data Locality impacted from unintended **mmrestripefile –l**

If the file is not set with WADFG (by policy or by mmchattr), **mmrestripefs -l** might break the data locality. The node running **mmrestripefile -l** is considered as the data writing node and all first replica of data is stored in the data writing node for an FPO-enabled storage pool.

The following sections describe the steps to check if your data locality is broken and how fix it if needed.

Check the data locality

This topic lists the steps to check the data locality for IBM Spectrum Scale.

Perform the following steps to check the data locality for IBM Spectrum Scale releases:
- For IBM Spectrum Scale 4.2.2.0 and earlier, run /usr/lpp/mmfs/samples/fpo/tsGetDataBlk.
- For IBM Spectrum Scale 4.2.2.x, run /usr/lpp/mmfs/samples/fpo/mmgetlocation.
- For IBM Spectrum Scale 4.2.3, **mmgetlocation** supports the -Y option.

Note: /usr/lpp/mmfs/samples/fpo/mmgetlocation is based on /usr/lpp/mmfs/samples/fpo/tsGetDataBlk. Ensure that GNU GCC is installed from Linux distro before invoking /usr/lpp/mmfs/samples/fpo/ mmgetlocation.

You can use /usr/lpp/mmfs/samples/fpo/mmgetlocation to query the block location of file.

You can refer the output from /usr/lpp/mmfs/samples/fpo/mmgetlocation about the options. You can run **/usr/lpp/mmfs/samples/fpo/mmgetlocation -f <absolute-file-path>** to get the block location of the <absolute-file-path>. Also, you can run **/usr/lpp/mmfs/samples/fpo/mmgetlocation -d <absolute-dir-path>** to get the block location summary of <absolute-dir-path>.

For IBM Spectrum Scale 4.2.2.x, run /usr/lpp/mmfs/samples/fpo/mmgetlocation.

The following is a sample output:
```
# /usr/lpp/mmfs/samples/fpo/mmgetlocation -f /sncfs/file1G

[FILE INFO]
-------------------------------------------------------------------------

blockSize 1024 KB
blockGroupFactor 128
metadataBlockSize 131072K
writeAffinityDepth 1
flags:
data replication: 2 max 2
storage pool name: fpodata
metadata replication: 2 max 2

Chunk 0 (offset 0) is located at disks: [ data_c8f2n04_sdg c8f2n04 ] [ data_c8f2n05_sdf c8f2n05 ]
...
Chunk 7 (offset 939524096) is located at disks: [ data_c8f2n04_sdg c8f2n04 ] [ data_c8f2n05_sdf c8f2n05 ]

[SUMMARY INFO]
---------------------------------------------------------------------------------------------------
Replica num Nodename TotalChunkst

Replica 1 : c8f2n04: Total : 8
Replica 2 : c8f2n05: Total : 8
[root@c8f2n04 fpo]#
```

The summary at the end of the output shows that, for the file /sncfs/file1G, 8 chunks of the first replica are located on the node c8f2n04. The 8 chunks of the second replica are located on the c8f2n05 node.

For IBM Spectrum Scale 4.2.2.0 and earlier, perform the following steps to get the block location of files.
```
cd /usr/lpp/mmfs/samples/fpo/
g++ -g -DGPFS_SNC_FILEMAP -o tsGetDataBlk -I/usr/lpp/mmfs/include/ tsGetDataBlk.C -L/usr/lpp/mmfs/lib/ -lgpfs
./tsGetDataBlk <filename>  -s 0 -f <data-pool-block-size * blockGroupFactor> -r 3
```

Check the output of the tsGetDataBlk program:

```
[root@gpfstest2 sncfs]# /usr/lpp/mmfs/samples/fpo/tsGetDataBlk /sncfs/test -r 3
File length: 1073741824, Block Size: 2097152
Parameters: startoffset:0, skipfactor: META_BLOCK, length: 1073741824, replicas 3
numReplicasReturned: 3, numBlksReturned: 4, META_BLOCK size: 268435456
Block 0 (offset 0) is located at disks:  2   4   6
Block 1 (offset 268435456) is located at disks:  2   4   6
Block 2 (offset 536870912) is located at disks:  2   4   6
Block 3 (offset 805306368) is located at disks:  2   4   6
```

In the above example, the block size of data pool is 2 Mbytes, the blockGroupFactor of the data pool is 128. So, the META_BLOCK (or chunk) size is 2MB * 128 = 256Mbytes. Each output line represents one chunk. For example, Block 0 in the above is located in the disks with disk id 2, 4 and 6 for 3 replica.

To know the node on which the three replicas of Block 0 are located, check the mapping between disk ID and nodes:

Check the mapping between disks and nodes by **mmlsdisk** (the 9th column is the disk id of NSD) and **mmlsnsd**:

```
[root@gpfstest2 sncfs]# mmlsdisk sncfs -L
disk          driver  sector    failure holds   holds            avail-            storage
name          type    size      group metadata data     status   ability  disk id pool        remarks
------------  ------- ------   ----------- -------- -----   -------  --------- ------- ---------   ---------
node1_sdb     nsd       512          1   Yes     No     ready    up             1 system      desc
node1_sdc     nsd       512      1,0,1   No      Yes    ready    up             2 datapool
node2_sda     nsd       512          1   Yes     No     ready    up             3 system
node2_sdb     nsd       512      2,0,1   No      Yes    ready    up             4 datapool
node6_sdb     nsd       512          2   Yes     No     ready    up             5 system      desc
node6_sdc     nsd       512      3,0,1   No      Yes    ready    up             6 datapool
node7_sdb     nsd       512          2   Yes     No     ready    up             7 system
node7_sdd     nsd       512      4,0,2   No      Yes    ready    up             8 datapool
node11_sdb    nsd       512          3   Yes     No     ready    up             9 system      desc
node11_sdd    nsd       512      1,1,1   No      Yes    ready    up            10 datapool    desc
node9_sdb     nsd       512          3   Yes     No     ready    up            11 system
node9_sdd     nsd       512      2,1,1   No      Yes    ready    up            12 datapool
node10_sdc    nsd       512          4   Yes     No     ready    up            13 system      desc
node10_sdf    nsd       512      3,1,1   No      Yes    ready    up            14 datapool
node12_sda    nsd       512          4   Yes     No     ready    up            15 system
node12_sdb    nsd       512      4,1,2   No      Yes    ready    up            16 datapool
[root@gpfstest2 sncfs]# mmlsnsd
 File system    Disk name    NSD servers
---------------------------------------------------------------------------
 sncfs          node1_sdb    gpfstest1.cn.ibm.com
 sncfs          node1_sdc    gpfstest1.cn.ibm.com
 sncfs          node2_sda    gpfstest2.cn.ibm.com
 sncfs          node2_sdb    gpfstest2.cn.ibm.com
 sncfs          node6_sdb    gpfstest6.cn.ibm.com
 sncfs          node6_sdc    gpfstest6.cn.ibm.com
 sncfs          node7_sdb    gpfstest7.cn.ibm.com
 sncfs          node7_sdd    gpfstest7.cn.ibm.com
 sncfs          node11_sdb   gpfstest11.cn.ibm.com
 sncfs          node11_sdd   gpfstest11.cn.ibm.com
 sncfs          node9_sdb    gpfstest9.cn.ibm.com
 sncfs          node9_sdd    gpfstest9.cn.ibm.com
 sncfs          node10_sdc   gpfstest10.cn.ibm.com
 sncfs          node10_sdf   gpfstest10.cn.ibm.com
 sncfs          node12_sda   gpfstest12.cn.ibm.com
 sncfs          node12_sdb   gpfstest12.cn.ibm.com
```

The three replicas of Block 0 are located in disk ID 2 (NSD name node1_sdc, node name is gpfstest1.cn.ibm.com), disk ID 4 (NSD name node2_sdb, node name is gpfstest2.cn.ibm.com), and disk ID 6 (NSD name node6_sdc, node name is gpfstest6.cn.ibm.com). Check each block of the file to see if the blocks are located correctly. If the blocks are not located correctly, fix the data locality.

mmgetlocation:

For IBM Spectrum Scale 4.2.3, mmgetlocation supports the -Y option.

Synopsis

```
mmgetlocation  {[-f filename] | [-d directory]}
               [-r {1|2|3|all}]
               [-b] [-L] [-l] [-Y] [--lessDetails]
               [-D [diskname,diskname,...]]
               [-N [nodename,nodename,...]]
```

Parameters

-f *filename*
   Specifies the file whose block location you want to query. It should be absolute file path. For one file, the system displays the block/chunk information and the file block summary information.

-d *directory*
   Specifies the directory whose block location you want to query. All files under <directory> will be checked and summarized together. **<directory>** must be the absolute directory path. The system displays one block summary for each file and one directory summary with the block information. The options **-f** and **-d** are exclusive.

   Note: The sub-directories under **<directory>** won't be checked.

[-r {1|2|3|all}]
   Specifies the replica that you want to query for the block location. 2 means replica 1 and replica 2. By default, the value is set to *all*.

-b The block location is considered as file system block or as FPO chunk(file system block size * blockGroupFactor). By default, the value is set to *no*.

-L Displays the detailed information (NSD ID and NSD failure group) of one block or chunk. This option impacts only the output of the block information for one file. It is not applicable when option **-d** is specified.

-l Lists the NSD and total replica number on the NSD in summary of file or directory.

-Y Displays headers and displays the output in a colon-separated fields format.

-D {*NSD***[,***NSD***...]}**
   Displays only the file block and chunk information and the summary of the specified NSDs.

-N {*Node***[,***Node***...]}**
   Displays only the file block and chunk information and the summary of the specified nodes.

--lessDetails
   Does not display the file summary of each file in **<directory>** when option **-d** is specified. If option **-f** is specified, does not display the block details.

Notes

1. Only tested over Linux.
2. Does not recursively process the subdirectories if option **-d** is specified.
3. For FPO, if both -D and -N are specified, the **-N** option must be with only one node because no two NSDs in FPO belong to the same node.
4. For **mmgetlocation -Y**, the system displays the output in the following formats:
   a. `mmgetlocation:fileSummary:filepath:blockSize:metadataBlockSize:dataReplica:metadataReplica:`
      `storagePoolName:allowWriteAffinity:writeAffinityDepth:blockGroupFactor:(-Y -L specified)`

b. `mmgetlocation:fileDataInfor:chunkIndex:offset:NSDName:NSDServer:diskID:failureGroup:`
   `reserved:NSDName:NSDServer:diskID:failureGroup:reserved:NSDName:NSDServer:diskID:failureGroup:`
   `reserved:` if there are 2 or 3 replicas, repeat "nsdName:nsdServer:diskID:failureGroup:reserved:"
   if the option "-L" is not specified, the value of "diskID" and "failureGroup" will be blank

   .

c. `mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:nsdName:blocks:(-l specified)`
   `mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:(-l not specified)` if there are
   more than 1 NSD for replica #, each one will be output as one line if the value of "nsdName" in
   one line is "all", that means, the option "-l" is not given.

d. `mmgetlocation:dirSummary:path:replicaIndex:nsdServer:nsdName:blocks:(-l specified)`

Note: If the value of **nsdName** in one line is *all*, the option **-l** is not given. So, for the option **-f**, the
output is:

a
b
c

For the option **-d**, the output is:

c for each file
d

Examples

```
l /usr/lpp/mmfs/samples/fpo/mmgetlocation -f /sncfs/file1G

[FILE /sncfs/file1G INFORMATION]
        FS_DATA_BLOCKSIZE : 1048576 (bytes)
        FS_META_DATA_BLOCKSIZE :  (bytes)
        FS_FILE_DATAREPLICA : 3
        FS_FILE_METADATAREPLICA : 3
        FS_FILE_STORAGEPOOLNAME : fpodata
        FS_FILE_ALLOWWRITEAFFINITY : yes
        FS_FILE_WRITEAFFINITYDEPTH : 1
        FS_FILE_BLOCKGROUPFACTOR : 128

chunk(s)# 0 (offset 0) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n02_sdc c3m3n02] [data_c3m3n04_sdc c3m3n04]
chunk(s)# 1 (offset 134217728) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n04_sdc c3m3n04] [data_c3m3n02_sdc c3m3n02]
chunk(s)# 2 (offset 268435456) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n02_sdc c3m3n02] [data_c3m3n04_sdc c3m3n04]
chunk(s)# 3 (offset 402653184) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n04_sdc c3m3n04] [data_c3m3n02_sdc c3m3n02]
chunk(s)# 4 (offset 536870912) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n02_sdc c3m3n02] [data_c3m3n04_sdc c3m3n04]
chunk(s)# 5 (offset 671088640) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n04_sdc c3m3n04] [data_c3m3n02_sdc c3m3n02]
chunk(s)# 6 (offset 805306368) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n02_sdc c3m3n02] [data_c3m3n04_sdc c3m3n04]
chunk(s)# 7 (offset 939524096) : [data_c3m3n03_sdd c3m3n03] [data_c3m3n04_sdc c3m3n04] [data_c3m3n02_sdc c3m3n02]

[FILE: /sncfs/file1G SUMMARY INFO]
replica1:
        c3m3n03: 8 chunk(s)
replica2:
        c3m3n04: 4 chunk(s)
        c3m3n02: 4 chunk(s)
replica3:
        c3m3n04: 4 chunk(s)
        c3m3n02: 4 chunk(s)

From the summary at the end of the output, you can know, for the file /sncfs/file1G,
8 chunks of the 1st replica are located on the node c3m3n03.
The 8 chunks of the 2nd replica are located on the node c3m3n04 and c3m3n02,
The 8 chunks of the 3nd replica are located on the node c3m3n04 and c3m3n02.

l /usr/lpp/mmfs/samples/fpo/mmgetlocation -d /sncfs/t2 -L -Y
mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:fileDataSummary:/sncfs/t2/_partition.lst:1:c3m3n04:1:
mmgetlocation:fileDataSummary:/sncfs/t2/_partition.lst:2::1:
mmgetlocation:fileDataSummary:/sncfs/t2/_partition.lst:3::1:
mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:fileDataSummary:/sncfs/t2/part-r-00000:1:c3m3n04:2:
mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:fileDataSummary:/sncfs/t2/part-r-00002:1:c3m3n04:2:
mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:fileDataSummary:/sncfs/t2/part-r-00001:1:c3m3n02:2:
mmgetlocation:dirDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:dirDataSummary:/sncfs/t2/:1:c3m3n04:5:
mmgetlocation:dirDataSummary:/sncfs/t2/:1:c3m3n02:2:
```

l /usr/lpp/mmfs/samples/fpo/mmgetlocation -f /sncfs/file1G -Y -L
mmgetlocation:fileSummary:filepath:blockSize:metadataBlockSize:dataReplica:metadataReplica:
storagePoolName:allowWriteAffinity:writeAffinityDepth:blockGroupFactor:
mmgetlocation:fileSummary:/sncfs/file1G:1048576::3:3:fpodata:yes:1:128:
mmgetlocation:fileDataInfor:chunkIndex:offset:NSDName:NSDServer:diskID:failureGroup:
reserved:NSDName:NSDServer:diskID:failureGroup:reserved:NSDName:NSDServer:diskID:failureGroup:reserved:
mmgetlocation:fileDataInfor:0:0):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n02_sdc:c3m3n02:3:1,0,
    0::data_c3m3n04_sdc:c3m3n04:9:2,0,0::
mmgetlocation:fileDataInfor:1:134217728):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n04_sdc:c3m3n04:9:2,0,
    0::data_c3m3n02_sdc:c3m3n02:3:1,0,0::
mmgetlocation:fileDataInfor:2:268435456):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n02_sdc:c3m3n02:3:1,0,
    0::data_c3m3n04_sdc:c3m3n04:9:2,0,0::
mmgetlocation:fileDataInfor:3:402653184):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n04_sdc:c3m3n04:9:2,0,
    0::data_c3m3n02_sdc:c3m3n02:3:1,0,0::
mmgetlocation:fileDataInfor:4:536870912):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n02_sdc:c3m3n02:3:1,0,
    0::data_c3m3n04_sdc:c3m3n04:9:2,0,0::
mmgetlocation:fileDataInfor:5:671088640):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n04_sdc:c3m3n04:9:2,0,
    0::data_c3m3n02_sdc:c3m3n02:3:1,0,0::
mmgetlocation:fileDataInfor:6:805306368):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n02_sdc:c3m3n02:3:1,0,
    0::data_c3m3n04_sdc:c3m3n04:9:2,0,0::
mmgetlocation:fileDataInfor:7:939524096):data_c3m3n03_sdd:c3m3n03:5:3,0,0::data_c3m3n04_sdc:c3m3n04:9:2,0,
    0::data_c3m3n02_sdc:c3m3n02:3:1,0,0::
mmgetlocation:fileDataSummary:path:replicaIndex:nsdServer:blocks:
mmgetlocation:fileDataSummary:/sncfs/file1G:1:c3m3n03:8:
mmgetlocation:fileDataSummary:/sncfs/file1G:2:c3m3n04:4:
mmgetlocation:fileDataSummary:/sncfs/file1G:2:c3m3n02:4:
mmgetlocation:fileDataSummary:/sncfs/file1G:3:c3m3n04:4:
mmgetlocation:fileDataSummary:/sncfs/file1G:3:c3m3n02:4:

l For IBM Spectrum Scale earlier than 4.2.2.0 perform the following steps to get block location of files.
1. cd /usr/lpp/mmfs/samples/fpo/
g++ -g -DGPFS_SNC_FILEMAP -o tsGetDataBlk -I/usr/lpp/mmfs/include/ tsGetDataBlk.C -L/usr/lpp/mmfs/lib/ -lgpfs
2. ./tsGetDataBlk <filename> -s 0 -f <data-pool-block-size * blockGroupFactor> -r 3
3. Check the output of the program tsGetDataBlk:
[root@gpfstest2 sncfs]# /usr/lpp/mmfs/samples/fpo/tsGetDataBlk /sncfs/test -r 3
File length: 1073741824, Block Size: 2097152
Parameters: startoffset:0, skipfactor: META_BLOCK, length: 1073741824, replicas 3
numReplicasReturned: 3, numBlksReturned: 4, META_BLOCK size: 268435456
Block 0 (offset 0) is located at disks: 2   4   6
Block 1 (offset 268435456) is located at disks: 2   4   6
Block 2 (offset 536870912) is located at disks: 2   4   6
Block 3 (offset 805306368) is located at disks: 2   4   6
4. In the above example, the block size of data pool is 2Mbytes, the blockGroupFactor of the
data pool is 128. So, the META_BLOCK (or chunk) size is 2MB * 128 = 256Mbytes. Each output line represents one chunk.
For example, Block 0 in the above is located in the disks with disk id 2, 4 and 6 for 3 replica.
In order to know the node on which the three replicas of Block 0 are located, check the mapping between disk ID and nodes:
Check the mapping between disks and nodes by mmlsdisk (the 9th column is the disk id of NSD) and mmlsnsd:
[root@gpfstest2 sncfs]# mmlsdisk sncfs —L

disk name	driver type	sector size	failure group	holds metadata	holds data	status	avail- ability	disk id	storage pool	remarks
node1_sdb	nsd	512	1	Yes	No	ready	up	1	system	desc
node1_sdc	nsd	512	1,0,1	No	Yes	ready	up	2	datapool	
node2_sda	nsd	512	1	Yes	No	ready	up	3	system	
node2_sdb	nsd	512	2,0,1	No	Yes	ready	up	4	datapool	
node6_sdb	nsd	512	2	Yes	No	ready	up	5	system	desc
node6_sdc	nsd	512	3,0,1	No	Yes	ready	up	6	datapool	
node7_sdb	nsd	512	2	Yes	No	ready	up	7	system	
node7_sdd	nsd	512	4,0,2	No	Yes	ready	up	8	datapool	
node11_sdb	nsd	512	3	Yes	No	ready	up	9	system	desc
node11_sdd	nsd	512	1,1,1	No	Yes	ready	up	10	datapool	desc
node9_sdb	nsd	512	3	Yes	No	ready	up	11	system	
node9_sdd	nsd	512	2,1,1	No	Yes	ready	up	12	datapool	
node10_sdc	nsd	512	4	Yes	No	ready	up	13	system	desc
node10_sdf	nsd	512	3,1,1	No	Yes	ready	up	14	datapool	
node12_sda	nsd	512	4	Yes	No	ready	up	15	system	
node12_sdb	nsd	512	4,1,2	No	Yes	ready	up	16	datapool	

[root@gpfstest2 sncfs]# mmlsnsd

File system	Disk name	NSD servers
sncfs	node1_sdb	gpfstest1.cn.ibm.com
sncfs	node1_sdc	gpfstest1.cn.ibm.com
sncfs	node2_sda	gpfstest2.cn.ibm.com
sncfs	node2_sdb	gpfstest2.cn.ibm.com
sncfs	node6_sdb	gpfstest6.cn.ibm.com
sncfs	node6_sdc	gpfstest6.cn.ibm.com
sncfs	node7_sdb	gpfstest7.cn.ibm.com
sncfs	node7_sdd	gpfstest7.cn.ibm.com
sncfs	node11_sdb	gpfstest11.cn.ibm.com
sncfs	node11_sdd	gpfstest11.cn.ibm.com
sncfs	node9_sdb	gpfstest9.cn.ibm.com
sncfs	node9_sdd	gpfstest9.cn.ibm.com
sncfs	node10_sdc	gpfstest10.cn.ibm.com

```
sncfs          node10_sdf    gpfstest10.cn.ibm.com
sncfs          node12_sda    gpfstest12.cn.ibm.com
sncfs          node12_sdb    gpfstest12.cn.ibm.com
The three replicas of Block 0 are located in disk id 2 (NSD name node1_sdc, node name is
gpfstest1.cn.ibm.com), disk id 4 (NSD name node2_sdb, node name is gpfstest2.cn.ibm.com),
and disk id 6 (NSD name node6_sdc, node name is gpfstest6.cn.ibm.com). Check each block of
the file to see if the blocks are located correctly. If all blocks are not located correctly,
fix the data locality
```

Data locality based copy

Synopsis

localityCopy *Device* {**-s** {[*Fileset*]: *Snapshot* | *srcDir* | *filePath*}
                    {**-t** *targetDir*} [**-l** | **-b**] [**-f**] [**-r**]
                    [**-a** | **-N** {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}]

Parameters

Device
> The device name of the file system to which the disks belong. File system names need not be fully-qualified. fs0 is as acceptable as /dev/fs0. This must be the first parameter.

{-s {[*Fileset*]: *Snapshot* | *srcDir* | *filePath*}
> Snapshot is the snapshot name. If *:Snapshot* is specified, the global snapshot is named Snapshot from Device. If there are more than 1 snapshots existing from *:Snapshot* or Snapshot, it will fail. Also, if it is fileset snapshot, ensure that the fileset is linked. *srcDir* is the source directory that is copied. The directory must exist in device. If the directory is the JunctionPath of one fileset, the fileset must be linked before running the script. *filePath* is the file path that will be copied.
>
> **Note:** Snapshot is the snapshot name. *srcDir* and *filePath* must be absolute path.

-t*targetDir*
> Specifies the target directory to which the files from the snapshot or the directory will be copied. *targetDir* must be absolute and must exist on the node that is running the command.

-l Only consider the locality if more than one node is involved. This might make some nodes busier than others. If there are active application jobs over the cluster and these jobs need enough network bandwidth, option **-l** makes the data copy consume as less as network bandwidth. When multiple nodes are specified with option **-N**, option **-l** might make the copy running over limited nodes and therefore take longer to finish data copy.

-b Considers the locality if more than one node is involved and distributes the copy tasks among all involved nodes.

> **Note:** The copy tasks are distributed at the file level (one file per copy task). The option **-l** and **-b** are exclusive. If either the option *-l* or **-b** is not specified, the option **-l** is true as default.

-f If the to-be-copied file exists under targetDir, it will be overwritten if the option *-f* is specified. Or, the file will be skipped.

-r When the option **-s** *{srcDir}* is specified, option **-r** will copy the files in recursive mode. For **-s** *{[Fileset]:Snapshot}*, option **-r** is always true.

-v Displays verbose information.

-a All nodes in the cluster are involved in copying tasks.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
> Directs a set of nodes to be involved in copying tasks. **-a** is the default if option **-N** is not specified.

Notes

1. If your file system mount point has special character, excluding +,-,_, it is not supported by this script.

2. If the file path contains special character, such as a blank character or a line break character, the file is not copied with warning.
3. When option **-a** or **-N** is specified, the file system for the **-t** *targetDir* must be mounted if it is from external NFS or another IBM Spectrum Scale file system.
4. Only copies the regular data file, does not copy link, special files.
5. If one file is not copied, the file is displayed and not copied again in the same invocation.
6. You must specify option **-s** with snapshot. For directory, the file list is not rescanned to detect any newly created files or subdirectories.

Data locality restoration
If the blocks of the file are not located as what you want, restore or change the locality of the file.

The IBM Spectrum Scale FPO provides interface for you to control all first replica of the blocks, all second replicas of the blocks, and all third replicas of the blocks in specific nodes. For example, you can have the first replica of all blocks located in a specific node so that the applications running over the node can read all data from local disks.

Note: The IBM Spectrum Scale FPO does not support the control of the location of only one or part of blocks. For example, you cannot control the location of block 1 or block 2 without changing the location of block 3.

Restoring the locality for files without WADFG:

This topic lists the steps to control the first replica of all blocks.
1. Check whether the file is configured with WADFG.
   ```
   mmlsattr -d -L /sncfs/test
   file name:            /sncfs/test
   metadata replication: 3 max 3
   data replication:     3 max 3
   immutable:            no
   appendOnly:           no
   flags:
   storage pool name:    datapool
   fileset name:         root
   snapshot name:
   Write Affinity Depth Failure Group(FG) Map for copy:1 1,0,1
   Write Affinity Depth Failure Group(FG) Map for copy:2 2,0,1
   Write Affinity Depth Failure Group(FG) Map for copy:3 3,0,1
   creation time:        Thu Mar 24 16:15:01 2016
   Misc attributes:      ARCHIVE
   Encrypted:            no
   gpfs.WADFG:           0x312C302C313B322C302C313B332C302C31
   ```
   If you see gpfs.WADFG (as per the preceding example) from the output of **mmlsattr**, the file is configured with WADFG, and, in this gpfs.WADFG case, follow the instructions in "Restoring the locality for files with WADFG" on page 550. If you do not see the gpfs.WADFG text, go to the step2.
2. Select the node to store all the blocks from the first replica of the data. One disk from this node is used to store the first replica of the file, assuming that this node has at least one local disk that serves the GPFS file system.

   In IBM Spectrum Scale 4.2.2.0 and later, **mmrestripefile -l** is optimized to reduce unnecessary data movement. For files with WAD=1, if the target node is from the same failure group as the current node holding the replica 1 of blocks, **mmrestripefile -l** does not move the second and third replica. If it is not, **mmrestripefile -l** moves three replicas of all the blocks
3. If you are using Spectrum Scale 4.1.1.0 or later:
   a. ssh to the target node selected in step 2
   b. run **mmrestripefile —l filename** for each filename to set the data locality for.
   If you are using Spectrum Scale 4.1.1.0 or earlier

a. ssh to the target node selected in step 2

b. **mmchdisk <fs-name> suspend -d "any-one-data-disk"**

c. **mmchdisk <fs-name> resume -d "any-one-data-disk"**

d. **mmrestripefile -b filename**

4. Check the data locality by running:

   `/usr/lpp/mmfs/samples/fpo/tsGetDataBlk /sncfs/test -r 3`

   The first replica of all blocks is located in the target node.

Restoring the locality for files with WADFG:

If you want to control the location of the first replica, second replica, and the third replica, set the WADFG attributes of the files via mmchattr. If you are using IBM Spectrum Scale 4.1.1.0 or earlier, perform these steps to restore the data locality.

1. Decide the location for data replica.

2. Run **mmchattr --write-affinity-failure-group** to set/update the new WADFG of the file Step.

   In IBM Spectrum Scale 4.2.2.0 and later, **mmrestripefile -l** is optimized to reduce unnecessary replica data movement. For example, the original WADFG is (1;2;3). If it is changed into (4;2;3), **mmrestripefile -l** moves only the first replica of all blocks. However, if it is changed into (4), **mmrestripefile -l** might move the second and third replica. Therefore, changing the original WADFG from (1;2;3) into (4;2;3) is better than changing it into (4).

3. If you are using IBM Spectrum Scale 4.1.1.0 or later, skip this step. Run **mmrestripefile -l filename** or **mmrestripefile -b filename**.

   In IBM Spectrum Scale 4.1.1.0 and later, the default option for **mmchattr** is -I yes, and the restripe function of **mmrestripefile -l** is performed when **mmchattr** is run.

4. Check the data locality.

Disk Replacement

This topic describes how to replace a disk.

In a production cluster, you can replace physically broken disks with new disks or replace the failed disks with new disks.

- If you have non functional disks from two failure groups for replica 3, restripe the file system to protect the data to avoid data loss from a third non functional disk from the third failure group.

- Replacing the disks is time-consuming because the whole inode space must be scanned and the IO traffic in the cluster is triggered. Therefore, schedule the disk replacement when the cluster is not busy.

The **mmrpldisk** command can be used to replace one disk in file system with a new disk and it can handle one disk in one invocation. If you want to replace only one disk, see **mmrpldisk** command.

Note: In FPO, sometimes **mmrpldisk** command does not migrate all data from the to-be-replaced disk to the newly added disk. This bug impacts IBM Spectrum Scale Release 3.5 and later. See the following example:

```
[root@c8f2n03 ~]# mmlsdisk sncfs -L
disk          driver   sector      failure holds   holds                    avail-          storage
name          type     size         group metadata data  status             ability disk id pool        remarks
------------ -------- ------ ----------- -------- ----- ------------- ------- ------- --------- --------
n03_0         nsd       512           1 Yes     Yes   ready             up            1 system
n03_1         nsd       512           1 Yes     Yes   ready             up            2 system      desc
n04_0         nsd       512       2,0,0 Yes     Yes   ready             up            3 system      desc
n04_1         nsd       512       2,0,0 Yes     Yes   ready             up            4 system
n05_1         nsd       512       4,0,0 No      Yes   ready             up            5 system      desc
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

```
[root@c8f2n03 ~]# /usr/lpp/mmfs/samples/fpo/tsGetDataBlk /sncfs/log -s 0
File length: 1073741824, Block Size: 1048576
Parameters: startoffset:0, skipfactor: META_BLOCK, length: 1073741824, replicas 0
numReplicasReturned: 2, numBlksReturned: 8, META_BLOCK size: 134217728
Block 0 (offset 0) is located at disks: 2    5
Block 1 (offset 134217728) is located at disks: 2    3
Block 2 (offset 268435456) is located at disks: 2    5
Block 3 (offset 402653184) is located at disks: 2    3
Block 4 (offset 536870912) is located at disks: 2    5
Block 5 (offset 671088640) is located at disks: 2    3
Block 6 (offset 805306368) is located at disks: 2    5
Block 7 (offset 939524096) is located at disks: 2    3

[root@c8f2n03 ~]# mmrpldisk sncfs n03_1 n03_4
[root@c8f2n03 ~]# mmlsdisk sncfs —L
disk          driver   sector     failure holds   holds            avail-          storage
name          type     size        group metadata data  status     ability disk id pool     remarks
------------  -------- ------ ----------- -------- ----- ----------- ------- ------- -------- --------
n03_0         nsd       512            1 Yes      Yes   ready       up            1  system   desc
n04_0         nsd       512        2,0,0 Yes      Yes   ready       up            3  system   desc
n04_1         nsd       512        2,0,0 Yes      Yes   ready       up            4  system
n05_1         nsd       512        4,0,0 No       Yes   ready       up            5  system   desc
n03_4         nsd       512            1 Yes      Yes   ready       up            6  system
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
[root@c8f2n03 ~]# /usr/lpp/mmfs/samples/fpo/tsGetDataBlk /sncfs/log -s 0
File length: 1073741824, Block Size: 1048576
Parameters: startoffset:0, skipfactor: META_BLOCK, length: 1073741824, replicas 0
numReplicasReturned: 2, numBlksReturned: 8, META_BLOCK size: 134217728
Block 0 (offset 0) is located at disks: 6    5
Block 1 (offset 134217728) is located at disks: 1    3
Block 2 (offset 268435456) is located at disks: 1    5
Block 3 (offset 402653184) is located at disks: 1    3
Block 4 (offset 536870912) is located at disks: 6    5
Block 5 (offset 671088640) is located at disks: 6    3
Block 6 (offset 805306368) is located at disks: 1    5
Block 7 (offset 939524096) is located at disks: 6    3
After replacing n03_1 with n03_4, part of data located in n03_1 are migrated into n03_4
and others are migrated into n03_0. Therefore, mmrpldisk doesn't mean copy data from the
to-be-replaced disks into new added disks. mmrpldisk might break the data locality and
you need to see the Section 9 to restore data locality if needed.
```

If you want to replace more than one disk, run the **mmrpldisk** command multiple times. The PIT job is triggered to scan the whole inode space to migrate the data to disks that are going to be replaced. The IO traffic is triggered and is time-consuming if you have to run the **mmrpldisk** command multiple times. To speed the replacement process, see the following sub sections to replace more than one disk in the file system.

Replace more than one active disks
This topic describes how to replace more than one active disk.

If you want to replace more than one disk used in file system, and if you have a lot of files or data in the file system, it will take long time to do this if you are using the **mmrpldisk** command for each disk.

If you have additional idle disk slots, you can plug new disks into these idle slots and run **mmcrnsd** to create new NSD disks against the disks that are to be added, run **mmadddisk** (without the option -r) to add the new disks into the file system and then **mmdeldisk** the disks that are to be replaced by using **mmdeldisk**.

Note: If you place new disks in the same failure group of the disks that are to be replaced, the above operations will maintain the data locality for the data from disks that are to be replaced. IBM Spectrum Scale keeps the data in the original failure group.

If you do not have additional idle disk slots, run the **mmdeldisk** command on the disks that are to be replaced, run **mmcrnsd** to create the NSD disks and run **mmadddisk** to add the NSD disks to the file system. You might have to run **mmrestripefs -b** to balance the file system but this breaks the data locality.

Replace more than one broken disks

If you want to replace more than one disk that are physically broken, you cannot read any data from these disks or write any data into these disks,

if the broken disks have been restriped they become emptied or non functional. Run the **mmdeldisk** command directly. Pull out the broken disks, pull in the new disks, run **mmcrnsd**, and then run **mmadddisk** to add them into the file system.

If the broken disks have not been restriped (then, it might be ready/down or ready/up), then take the following steps:

1. Disable auto recovery temporarily (refer the section 2.1, step 2)
2. Pull out the broken disks directly. You could run **mmlsnsd -X** to check what these pulled-out disks will be like: node7_sdn C0A80A0756FBAA89 - - gpfstest7.cn.ibm.com (not found) server node
3. Pull in the new disks.
4. **mmcrnsd** for the new disks (take new NSD name)
5. **mmadddisk <fs-name> -F <new-nsd-file from step4>**
6. To delete the broken disks from the file system, see *Disk media failure* in *IBM Spectrum Scale: Problem Determination Guide*.

Auto recovery

The FPO-enabled/disabled storage pool over internal disks are subject to frequent node and disk failures because of the commodity hardware used in IBM Spectrum Scale clusters.

IBM Spectrum Scale auto recovery feature is designed to handle random but routine node and disk failures without requiring manual intervention. However, auto recovery cannot cover all catastrophic outages involving large number of nodes and disks at once. Administrator assessment of the situation and judgment is required to determine the cluster recovery action.

Note:

Following are some important recommendations:
- Once the disks are suspended by the auto recovery, these disks must be manually resumed by the administrator so that the disks can be used by the file system. When the owning node is determined to be the healthy again, **mmchdisk resume** command must be run.
- If extended outages (days and weeks) are expected, it is recommended to remove that node and all associated disks from the cluster to avoid this outage from affecting subsequent recovery actions.
- If the failed disk is meta disk, during auto recovery, GPFS will try to suspend the failed disk using the **mmchdisk <file-system>** command. If the remaining failure groups of meta or data disks is less than the value of -r/-m, this will make **mmchdisk <file-system>** suspend/fail, and therefore auto recovery will not take further actions.

Failure and recovery

There are two main failures to consider for FPO environments.

1. Node failure and outages: These outages include reboot, kernel crash and hang and can last long. When a node is inaccessible, all the associated disks also become inaccessible.
2. Disk failures: These failures include disk failures, hard IO errors and are generally triggered by hardware failures and affect specific disks.

IBM Spectrum Scale recovery actions are enabled by setting the `restripeOnDiskFailure` configuration option to yes. When this option is enabled, auto recovery leverages the IBM Spectrum Scale event callback mechanism to trigger necessary actions to perform recovery actions. Specifically, the following system callbacks are installed when `restripeOnDiskFailure=yes`.

- **event = diskFailure action: /usr/lpp/mmfs/bin/mmcommon recoverFailedDisk %fsName %diskName**
- **event = nodeJoin action: /usr/lpp/mmfs/bin/mmcommon restartDownDisks %myNode %clusterManager %eventNode**
- **event = nodeLeave action: /usr/lpp/mmfs/bin/mmcommon stopFailedDisk %myNode %clusterManager %eventNode**

diskFailure Event

This event is triggered when a disk I/O operation fails. Upon I/O failure, IBM Spectrum Scale marks the disk from read/up to ready/down. This I/O failure can also be caused by a node, because all disks connected by the node become unavailable, or a disk failure.

The disk state is ready/down.

Recovery process

1. Perform simple checks, such as fpo pool and replication >1.
2. Check the **maxDownDisksForRecovery** (default16), **maxFailedNodesForRecovery** (default 3). Abort if the limit is exceeded. Note that these limits can be changed by using the configuration parameters.
3. If the number of failed FGs is less than 2, wait until **dataDiskWaitTimeForRecovery** (default 3600/2400) expires, otherwise wait for **minDiskWaitTimeForRecovery** (default 1800 sec) to expedite recovery due to increased risk.
4. If the available FGs for metadata is less than three, no action is taken because recovery cannot be performed due to the metadata outage.
5. After the recovery wait period has passed, recheck the node and disk availability status to ensure that recovery actions are taken.
6. Suspend all the failed and unavailable disks by running the **tschdisk suspend** command.
7. Restripe the data. If a previous restripe process is running, stop it and start a new process.
8. At successful completion, disks will be in suspended/down or suspended/up if the node is recovered during the restripe.

nodeJoin Event

This event is triggered when a node joins the cluster after a node reboot or rejoined after losing membership to the cluster or getting started after an extended outage. Scope of the recovery is all file systems to which the node disks might belong to. In most case, the disk state can be ready/up if no I/O operation has been performed or ready/down. However, based on the prior events, the state could vary to suspended/down or unrecovered/recovering.

Recovery process

1. Perform simple checks on the disks assigned to the file systems.
2. Check if a **tschdisk** start is already running from a prior event. Kill the process to include disks from the current nodes.
3. Start all disks on all nodes by running: **tschdisk start -a** to optimize recovery time. This command requires all nodes in the cluster to be functioning in order to access all the disks in the file system.

4. Start All down disks on all Active nodes by running: `tschdisk start -F<file containing disk list>`.
5. After successful completion, all disks must be in the ready/up or the suspended/up state if the node is being recovered from a long outage.

If a new diskFailure event is triggered while **tschdisk** start is in progress, the disks will not be restored to the Up state until the node joins the cluster and triggers a nodeJoin event.

nodeLeave Event

This event is triggered when a node leaves the cluster, is expelled, or shut down.

The processing of this event is similar to the diskFailure event, except that disks may not already be marked as Down when this event is received. Note that a diskFailure event can still be generated based on an I/O activity in the cluster. If it is generated, no action will be taken by the diskFailure event handler if the owning node is also down, thereby allowing the nodeLeave event to control the recovery. In most cases, the disk state could be ready/up if no I/O operation has been performed or ready/down. However, based on prior events the state could be suspended/down or unrecovered/recovering.

Recovery process

1. Wait for the specified duration to give the failed nodes a chance to recover.
2. Check the Down nodes count, Down disks count and available data and metadata FG count to check against the maximum limit.
3. Build a list of disk to act upon, ignoring suspended, empty, to be emptied.
4. Run **tsrestripefs** to restore replica count to the stated values.
5. After successful completion, disks can be in the suspended/down state or no action may be taken if the nodeJoin event is triggered within the **recoveryWaitPeriod**.

QoS support for autorecovery

When QoS is configured, the autorecovery process runs in a QoS class.

To get QoS support for autorecovery, you must enable QoS and assign IOPS to the **maintenance** and **other** classes of the storage pools that you want autorecovery to restore. For more information, see "Setting the Quality of Service for I/O operations (QoS)" on page 125.

In IBM Spectrum Scale v4.2.1.x and v4.2.2.x, the autorecovery process always runs in the QoS **maintenance** class. If you assign a smaller share of IOPS to the **maintenance** class, this setting ensures that autorecovery does not compete with normal processes for I/O operations.

In IBM Spectrum Scale v4.2.3 and later, the autorecovery process runs in the QoS **maintenance** class only if one replica is lost. If more than one replica is lost, the autorecovery process runs in the QoS **other** class so that it completes faster.

Restrictions

An FPO environment includes restrictions.

The following restrictions apply:
- Storage pool properties can be set only when the pool is created and cannot be changed later.
- All disks in an FPO pool must be assigned an explicit failure group.
- All disks in an FPO pool must have exactly one NSD server associated with them.
- All disks in an FPO pool that share an NSD server must belong to the same failure group.
- When replacing a disk in an FPO pool, the old and new disks must have the same NSD server.

- Disks must be removed from the file system before NSD servers can be changed.
- FPO is not supported on Windows.

There might be additional limitations and restrictions. For the latest support information, see the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Chapter 35. Encryption

GPFS provides support for file encryption that ensures both secure storage and secure deletion of data. GPFS manages encryption through the use of encryption keys and encryption policies.

Note: GPFS encryption is only available with IBM Spectrum Scale Advanced Edition or IBM Spectrum Scale Data Management Edition. The file system must be at GPFS V4.1 or later. Encryption is supported in:
- Multicluster environments (provided that the remote nodes have their own `/var/mmfs/etc/RKM.conf` files and access to the remote key management servers. For more information, see "Encryption keys.")
- FPO environments

Secure storage uses encryption to make data unreadable to anyone who does not possess the necessary encryption keys. The data is encrypted while "at rest" (on disk) and is decrypted on the way to the reader. Only data, not metadata, is encrypted.

GPFS encryption can protect against attacks targeting the disks (for example, theft or acquisition of improperly discarded disks) as well as attacks performed by unprivileged users of a GPFS node in a multi-tenant cluster (that is, a cluster that stores data belonging to multiple administrative entities called tenants). However, it cannot protect against deliberate malicious acts by a cluster administrator.

Secure data deletion leverages encryption and key management to guarantee erasure of files beyond the physical and logical limitations of normal deletion operations. If data is encrypted, and the master key (or keys) required to decrypt it have been deleted from the key server, that data is effectively no longer retrievable. See "Encryption keys."

Important: Encryption should not be viewed as a substitute for using file permissions to control user access.

Encryption keys

GPFS uses the following types of encryption keys:

master encryption key (MEK)
> An MEK is used to encrypt file encryption keys.
>
> MEKs are stored in remote key management (RKM) servers and are cached by GPFS components. GPFS receives information about the RKM servers in a separate `/var/mmfs/etc/RKM.conf` configuration file. Encryption rules present in the encryption policy define which MEKs should be used, and the `/var/mmfs/etc/RKM.conf` file provides a means of accessing those keys. The `/var/mmfs/etc/RKM.conf` also specifies how to access RKMs containing MEKs used to encrypt files created under previous encryption policies.
>
> An MEK is identified with a unique *Keyname* that combines the name of the key and the RKM server on which it resides. See "Encryption policy rules" on page 558 for *Keyname* format.

file encryption key (FEK)
> An FEK is used to encrypt sectors of an individual file. It is a unique key that is randomly generated when the file is created. For protection, it is encrypted (or "wrapped") with one or more MEKs and stored in the `gpfs.Encryption` extended attribute of the file.
>
> A wrapped FEK cannot be decoded without access to the MEK (or MEKs) used to wrap it. Therefore, a wrapped FEK is useless to an attacker and does not require any special handling at

object deletion time. If necessary, an FEK can be rewrapped using a new set of MEKs to allow for operations like MEK expiration and rotation, compromised key removal, and data expiration.

Note: If an encryption policy specifies that an FEK be wrapped multiple times, only one of the wrapped-FEK instances needs to be unwrapped for the file to be accessible.

Encryption policies

GPFS uses encryption policies to manage aspects of how file encryption is to be implemented, including the following:
- which files are to be encrypted
- which algorithm is to be used for the encryption
- which MEK (or MEKs) are to be used to wrap the FEK of a file

Encryption policies are configured using the **mmchpolicy** command and are applied at file creation time. When a file is created, encryption rules are traversed in order until one of the following occurs:
- The last rule is reached.
- The maximum number of **SET ENCRYPTION** rules that can be matched (eight) is reached.
- An **ENCRYPTION EXCLUDE** rule is matched.

If the file matches at least one **SET ENCRYPTION** rule, an FEK is generated and used to encrypt its contents. The FEK is wrapped once for each policy it matches, resulting in one or more versions of the encrypted FEK being stored in the `gpfs.Encryption` extended attribute of the file.

Notes:
1. When an encryption policy is changed, the changes apply only to the encryption of subsequently created files.
2. Encryption policies are defined on a per–file system basis by a system administrator. Once the encryption policies are put in place, they may result in files in different filesets or with different names being encrypted differently.

Encryption policy rules

GPFS provides the following rules with which you can specify encryption policies:

ENCRYPTION IS

This rule is used to specify how a file is to be encrypted and how the FEK is to be wrapped.

The syntax of the **ENCRYPTION IS** rule is:

```
RULE 'RuleName' ENCRYPTION 'EncryptionSpecificationName' IS
    ALGO 'EncParamString'
    COMBINE 'CombineParamString'
    WRAP 'WrapParamString'
    KEYS('Keyname'[, 'Keyname', ... ])
```

where:

ALGO *EncParamString*

specifies the encryption parameter string, which defines the following:
- encryption algorithm
- key length
- mode of operation
- key derivation function

The following encryption parameter strings are valid:

Table 44. Valid EncParamString values

Value	Description
AES:128:XTS:FEK:HMACSHA512	Encrypt the file with AES in XTS mode. The FEK is 128 bits long and is preprocessed using HMAC with SHA-512.
AES:256:XTS:FEK:HMACSHA512	Encrypt the file with AES in XTS mode. The FEK is 256 bits long and is preprocessed using HMAC with SHA-512.
AES:128:CBC:FEK:HMACSHA512	Encrypt the file with AES in CBC mode. The FEK is 128 bits long and is preprocessed using HMAC with SHA-512.
AES:192:CBC:FEK:HMACSHA512	Encrypt the file with AES in CBC mode. The FEK is 192 bits long and is preprocessed using HMAC with SHA-512.
AES:256:CBC:FEK:HMACSHA512	Encrypt the file with AES in CBC mode. The FEK is 256 bits long and is preprocessed using HMAC with SHA-512.

COMBINE *CombineParamString*
> specifies a string that defines the mode to be used to combine MEKs specified by the **KEY** statement.
>
> The following combine parameter string values are valid:

Table 45. Valid combine parameter string values

Value	Description
XORHMACSHA512	Combine MEKs with a round of XOR followed by a round of HMAC with SHA-512.
XOR	Combine MEKs with a round of XOR.

WRAP *WrapParamString*
> specifies a string that defines the encryption algorithm and the wrapping mode to be used to wrap the FEK.
>
> The following wrapping parameter string values are valid:

Table 46. Valid wrapping parameter string values.

Value	Description
AES:KWRAP	Use AES key wrap to wrap the FEK.
AES:CBCIV	Use AES in CBC-IV mode to wrap the FEK.

KEYS ('*Keyname***'[, '***Keyname***', ...])**
> specifies one or more keys to be applied. Each *Keyname* is a unique identifier that combines the name of the key and the RKM server on which it resides. The format for *Keyname* is:
>
> *KeyId***:***RkmId*
>
> where
>
> *KeyId*
> > An internal identifier that uniquely identifies the key inside the RKM. Valid characters for *KeyId* are the following: 'A' through 'Z'; 'a' through 'z'; '0' through '9'; and '-' (hyphen). The minimum length of *KeyId* is one character; the maximum length is 42 characters.

RkmId

The identifier of the `/var/mmfs/etc/RKM.conf` entry for the RKM that manages the key. An RKM ID must be unique within the cluster, must be 1-21 characters in length, and can contain only the characters a - z, A - Z, 0 - 9, or underscore (_). The first character cannot be a numeral.

Notes:

1. The maximum number of keys you can specify with the **ENCRYPTION IS** rule is eight.
2. The number of keys that can be used to encrypt a single file is permanently limited by the inode size of the file system.
3. You cannot specify the same key more than once in a given **ENCRYPTION IS** rule. Also, do not specify keys with identical values in an **ENCRYPTION IS** rule. Specifying the same key or identically-valued keys could result in a security breach for your data.

SET ENCRYPTION

The **SET ENCRYPTION** rule is similar to the **SET POOL** rule. If more than one such rule is present, all **SET ENCRYPTION** rules are considered and the FEK is wrapped once for each of the rules that apply (up to the maximum of eight). As mentioned in "Encryption keys" on page 557, if an FEK is wrapped multiple times, only one of the wrapped-FEK instances needs to be unwrapped for the file to be accessed.

If no **SET ENCRYPTION** rule is applicable when a file is created, the file is not encrypted. The syntax of the **SET ENCRYPTION** rule is:

```
RULE 'RuleName' SET ENCRYPTION 'EncryptionSpecificationName'[, 'EncryptionSpecificationName',...]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [WHERE SqlExpression]
```

where:

EncryptionSpecificationName

is the name of a specification defined by an **ENCRYPTION IS** rule.

To stop traversing policy rules at a certain point and encrypt using only those rules that have matched up to that point, use the **SET ENCRYPTION EXCLUDE** rule:

```
RULE ['RuleName']  SET ENCRYPTION EXCLUDE
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [WHERE SqlExpression]
```

Note: Encryption policies do not support the **ACTION** clause.

Default encryption parameters

To simplify policy management, GPFS accepts the special default value `'DEFAULTNISTSP800131A'` as the **ALGO** parameter string.

For example, this policy statement:

```
RULE 'somerule' ENCRYPTION 'somename' IS
    ALGO 'DEFAULTNISTSP800131A'
    KEYS('KEY-2f1f7700-de74-4e55-a9be-bee49c5b3af8:RKMKMIP3')
```

corresponds to this:

```
RULE 'somerule' ENCRYPTION 'somename' IS
    ALGO 'AES:256:XTS:FEK:HMACSHA512'
    COMBINE 'XORHMACSHA512'
    WRAP 'AES:KWRAP'
    KEYS('KEY-2f1f7700-de74-4e55-a9be-bee49c5b3af8:RKMKMIP3')
```

Note: When this special **ALGO** default value is set as the **ALGO** *EncParamString*, neither **COMBINE** nor **WRAP** should be specified.

In the example above, RKMKMIP3 is the RKM ID, which is defined in the RKM stanza in the RKM.conf file.

Example of an encryption policy

This is an example of an encryption policy:

```
RULE 'myEncRule1' ENCRYPTION 'E1' IS
        ALGO 'DEFAULTNISTSP800131A'
        KEYS('1:RKM_1', '2:RKM_2')

RULE 'myEncRule2' ENCRYPTION 'E2' IS
        ALGO 'AES:256:XTS:FEK:HMACSHA512'
        COMBINE 'XOR'
        WRAP 'AES:KWRAP'
        KEYS('3:RKM_1')

RULE 'myEncRule3' ENCRYPTION 'E3' IS
        ALGO 'AES:128:CBC:FEK:HMACSHA512'
        COMBINE 'XORHMACSHA512'
        WRAP 'AES:CBCIV'
        KEYS('4:RKM_2')

RULE 'Do not encrypt files with extension enc4'
        SET ENCRYPTION EXCLUDE
        FOR FILESET('fs1')
        WHERE NAME LIKE '%.enc4'

RULE 'Encrypt files with extension enc1 with rule E1'
        SET ENCRYPTION 'E1'
        FOR FILESET('fs1')
        WHERE NAME LIKE '%.enc1'

RULE 'Encrypt files with extension enc2 with rule E2'
        SET ENCRYPTION 'E2'
        FOR FILESET('fs1')
        WHERE NAME LIKE '%.enc2'

RULE 'Encrypt files with extension enc* with rule E3'
        SET ENCRYPTION 'E3'
        FOR FILESET('fs1')
        WHERE NAME LIKE '%.enc%'
```

Note:

In this example encryption policy:

- All files in fileset fs1 are treated as follows:
  - If the extension is equal to enc4, the file is not encrypted. This happens because the ENCRYPTION EXCLUDE rule is matched first, stopping the traversal of the remaining rules before any additional matches can be made.
  - If the extension is equal to enc1, the file is encrypted with a 256-bit FEK, using AES in XTS mode; the FEK is preprocessed with HMAC with SHA-512, and the FEK is then wrapped twice:
    - once with AES key wrap, with keys 1:RKM_1 and 2:RKM_2 combined via one round of XOR followed by one round of HMAC with SHA-512
    - once with AES in CBC-IV mode using key 4:RKM_2

    This happens because both rules E1 and E3 apply, since extension enc1 matches both %.enc1 and %.enc%. Note that the encryption algorithms specified by rule E1, which grant a stronger security than those of rule E3, are chosen and applied.
  - If the extension is equal to enc2, the file is encrypted with a 256-bit FEK, using AES in XTS mode; the FEK is preprocessed with HMAC with SHA-512; and the FEK is then wrapped twice:

- once with AES key wrap using key `3:RKM_1`
- once with AES in CBC-IV mode using key `4:RKM_2`

This happens because both rules E2 and E3 apply, since extension `enc2` matches both `%.enc2` and `%.enc%`.

– If the extension is equal to `enc3`, the file is encrypted with a 128-bit FEK, using AES in CBC mode; the FEK is preprocessed with HMAC with SHA-512; and the FEK is then wrapped once with AES in CBC-IV mode using key `4:RKM_2`.

This happens because only rule E3 applies, since extension `enc3` only matches `%.enc%`.

- A GPFS node with access to both keys `1:RKM_1` and `2:RKM_2` or to key `4:RKM_2` can access a file with extension `enc1`.
- A GPFS node with access to key `3:RKM_1` or to key `4:RKM_2` can access a file with extension `enc2`.
- A GPFS node with access to key `4:RKM_2` can access a file with extension `enc3`.
- No key is required to access a file with extension `enc4`.
- A file with extension `enc1` is securely deleted when either key `1:RKM_1` or `2:RKM_2`, and key `4:RKM_2` are destroyed in their respective RKMs (and their cached copies have been flushed).
- A file with extension `enc2` is securely deleted when key `3:RKM_1` and key `4:RKM_2` are destroyed in their respective RKMs (and their cached copies have been flushed).
- A file with extension `enc3` is securely deleted when key `4:RKM_2` is destroyed in its respective RKM (and its cached copies have been flushed).
- Once created, a file may not be encrypted with more MEKs, only with different MEKs using the **REWRAP** rule.

Rewrapping policies

Rewrapping policies are used to change the way a set of FEKs is encrypted; that is, to change the set of MEKs that wrap the FEKs of those files. Rewrapping applies only to files that are already encrypted, and the rewrapping operation acts only on the `gpfs.Encryption` EA of the files. Rewrapping is done by using the **mmapplypolicy** command to apply a set of policy rules containing one or more `CHANGE ENCRYPTION KEYS` rules. These rules have the form:

```
RULE 'ruleName' CHANGE ENCRYPTION KEYS FROM 'Keyname_1' to 'Keyname_2'
[FROM POOL 'poolName']
  [FOR FILESET(...)]
  [SHOW(...)]
  [WHERE ... ]
```

where:

- *Keyname_1* is the unique identifier of the MEK to be replaced. (See "Encryption policy rules" on page 558 for *Keyname* format.)
- *Keyname_2* is the unique identifier of the new MEK, which will replace the old MEK identified by *Keyname_1*.
- The `FOR FILESET` and `WHERE` clauses narrow down the set of affected files.

Both *Keyname_1* and *Keyname_2* are listed, and only the files that currently use *Keyname_1* will have their FEKs rewrapped with *Keyname_2*. Files that do not currently use *Keyname_1* are not affected by the operation.

Notes:

1. Only the *first* matching **CHANGE ENCRYPTION KEYS** rule will be applied to each file. The rule will rewrap each wrapped version of the FEK that was encrypted with the MEK in the **CHANGE ENCRYPTION KEYS** rule.
2. The same MEK cannot be used more than once in a particular wrapping of the FEK.

Preparation for encryption

Preparing for encryption includes verifying the version of IBM Spectrum Scale, installing a remote encryption key server, preparing the cluster, and preparing the encryption key server back ends.

Terms defined

You should be familiar with the following terms:

device group
> See *tenant*.

file encryption key
> A *file encryption key (FEK)* is an encryption key that a key client uses to encrypt a data file. See "Encryption keys" on page 557.

key client
> A *key client* is a computer system, such as an IBM Spectrum Scale node, that retrieves master encryption keys from a key server.

key server
> A *key server*, also known as a *Remote Key Management (RKM) server*, is a server that provides master encryption keys for key clients. Examples of key server software products are IBM Security Key Lifecycle Manager (SKLM) and Vormetric Data Security Manager (DSM).

master encryption key
> A *master encryption key (MEK)* is an encryption key that a key client uses to encrypt a file encryption key.

tenant A *tenant* is an entity on a key server that contains master encryption keys and certificates. In the Vormetric Data Security Manager (DSM), a tenant is called a *device group*.

Required software: IBM Spectrum Scale

The following table lists the versions of IBM Spectrum Scale that support encryption and the encryption setup methods:

Table 47. Required version of IBM Spectrum Scale

IBM software	Version	Encryption setup
IBM Spectrum Scale • Advanced Edition • Data Management Edition	V4.1 or later	• Regular setup • Regular setup with certificate chain
	V4.2.1 or later	Simplified setup

Required software: Remote Key Management (RKM) server

The next table shows the RKM server software that IBM Spectrum Scale supports.

Table 48. Remote Key Management servers

RKM server	Version	Type of encryption setup
IBM Security Key Lifecycle Manager (SKLM)	V2.5.0.1 or later	• Regular setup • Regular setup with certificate chain
IBM Security Key Lifecycle Manager (SKLM)	V2.5.0.4 or later	Simplified setup
Vormetric Data Security Manager (DSM)	V5.2.3 or later	Regular setup

Note: IBM SKLM and Vormetric DSM have a complete implementation of the Key Management Interoperability Protocol (KMIP) standard of the Organization for the Advancement of Structured Information Standards (OASIS). IBM Spectrum Scale nodes use the KMIP protocol to retrieve keys from SKLM and Vormetric Data Security Manager (DSM) servers.

Preparing your cluster for encryption

Follow these steps:
1. Verify the following items in your IBM Spectrum Scale cluster:
   - The cluster is running the correct version of IBM Spectrum Scale and the correct version of a supported RKM server. These versions are listed in Table 47 on page 563 and Table 48.
   - The file system daemon is running.
2. Ensure that the following packages are installed:
   - gpfs.gskit
   - gpfs.crypto
3. Set up an IBM Spectrum Scale file system on the cluster. The version of the file system must be IBM Spectrum Scale Release 4.1 or later. Configure the following features on the file system:
   a. Create the file system with the inode size of 4 KB. This size is the recommended minimum size. The 4 KB inode size is recommended to accommodate the gpfs.Encryption extended attribute that is assigned to each encrypted file at file creation time. This extended attribute contains one or more wrapped file encryption keys (FEKs) so it can potentially grow large. For more information, see "Encryption policies" on page 558.
   b. Enable fast extended attributes. This setting is the default for a newly created file system if you are running V4.1 or later. To verify that fast extended attributes are enabled, issue a command like the following one:

      ```
      mmlsfs <Device> --fastea
      ```

      where *<Device>* is the name of the file system. However, if your file system was migrated from an earlier level, you might need to enter the following command to add support for fast extended attributes:

      ```
      mmmigrate <Device> --fastea
      ```

      where *<Device>* is the name of the file system. For more information, see the topic *Completing the migration to a new level of IBM Spectrum Scale* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide.*

Preparing the remote key management (RKM) server

The preparation of the RKM server depends on the RKM server product that you select and the encryption setup that you plan to follow. For more information, see the help topic in the following list that describes the setup of your RKM server:

- "Simplified setup: Using SKLM with a self-signed certificate" on page 568
- "Regular setup: Using SKLM with a self-signed certificate" on page 597
- "Configuring encryption with the Vormetric DSM key server" on page 616

RKM back ends

An RKM back end defines a connection between a local key client, a remote key tenant, and an RKM server. Each RKM back end is described in an RKM stanza in an `RKM.conf` file on each node that is configured for encryption.

By controlling the contents of the `RKM.conf` file, the cluster administrator can control which client nodes have access to master encryption keys (MEKs). For example, the same RKM server can be given two different names in `/var/mmfs/etc/RKM.conf` stanzas. Then, the administrator can partition a set of MEKs hosted on a single RKM server into separate subsets of MEKs. These subsets of MEKs might belong to subsets of the nodes of the cluster.

Because the master encryption keys (MEK) are cached in memory, some short-term outages while accessing a key server might not cause issues. However, failure to retrieve the keys might result in errors while creating, opening, reading, or writing files. Although the keys are cached, they are periodically retrieved from the key server to ensure their validity.

To ensure that MEKs are always available, it is a good practice to set up multiple key servers in a high-availability configuration. See the subtopic "Identifying multiple RKM back ends in a high-availability configuration" on page 566.

Note: If you are using the simplified setup, then the **mmkeyserv** command manages its own `RKM.conf` file and updates it automatically. This includes adding any backup servers for High Availability and other key retrieval properties. The `RKM.conf` file that the **mmkeyserv** command manages is in the `/var/mmfs/ssl/keyServ` directory.

The `RKM.conf` file and the RKM stanza

The management of the `RKM.conf` file and its stanzas depends on the setup:

- In the simplified setup, the **mmkeyserv** command manages its own `RKM.conf` file and updates it automatically. This management includes adding any backup servers for High Availability and other key retrieval properties.
- In the regular setup and the Vormetric setup, you must manage the RKM.conf file and its contents.

The location of the `RKM.conf` file also depends on the setup:

Table 49. The RKM.conf file

Setup	Location of the `RKM.conf` file
Simplified setup	`/var/mmfs/ssl/keyServ/RKM.conf`
Regular setup	`/var/mmfs/etc/RKM.conf`
Vormetric DSM setup	`/var/mmfs/etc/RKM.conf`

The length of the RKM.conf file cannot exceed 1 MiB. No limit is set on the number of RKM stanzas, if the length limit is not exceeded.

After the file system is configured with encryption policy rules, the file system is considered encrypted. From that point on, each node that has access to that file system must have an `RKM.conf` file present. Otherwise, the file system might not be mounted or might become unmounted.

Each RKM stanza in the `RKM.conf` file describes a connection between a local key client, a remote tenant, and an RKM server. The following code block shows the structure of an RKM stanza:

```
RKM ID {
    type = ISKLM
    kmipServerUri = tls://host:port
    keyStore = /PathToKeyStoreFile
    passphrase = Password
    clientCertLabel = LabelName
    tenantName = NameOfTenant
    [connectionTimeout = ConnectionTimeout]
    [connectionAttempts = ConnectionAttempts]
    [retrySleep = RetrySleepUsec]
}
```

where the terms of the stanza have the following meanings:

RKM ID
> The name of the stanza.

type ISKLM for the regular setup and the simplified setup. KMIP for the Vormetric DSM setup.

kmipServerUri
> The DNS name or IP address of the SKLM or DSM server and the KMIP SSL port.

keyStore
> The path and name of the client keystore.

passphrase
> The password of the client keystore and client certificate.

clientCertLabel
> The label of the client certificate in the client keystore.

tenantName
> The name of the tenant or device group.

connectionTimeout
> The connection timeout, in seconds. The default is 60 seconds. The valid range is 1 - 120 seconds.

connectionAttempts
> The number of connection attempts. The default is 3 attempts. The valid range is 1 - 10.

retrySleep
> The retry sleep time, in microseconds. The default is 100,000 (0.1 seconds). The valid range is 1 - 10,000,000 microseconds.

Identifying multiple RKM back ends in a high-availability configuration

The SKLM supports automated replication across multiple nodes for high-availability deployments. To identify multiple RKM back ends in a high-availability configuration, specify any of the following optional parameters:

```
rkmname3 {
...
    kmipServerUri2 = tls://host:port     # TLS connection to clone number 1 to host on port
    kmipServerUri3 = tls://host:port     # TLS connection to clone number 2 to host on port
    kmipServerUri4 = tls://host:port     # TLS connection to clone number 3 to host on port
    kmipServerUri5 = tls://host:port     # TLS connection to clone number 4 to host on port
    kmipServerUri6 = tls://host:port     # TLS connection to clone number 5 to host on port
...
}
```

If at least one backup is configured, whenever key retrieval from the master fails, IBM Spectrum Scale looks in each backup until it finds the MEK. The addition of the URIs for the clone servers is the only required change within IBM Spectrum Scale. All other configuration parameters (certificates, keys, node, and tenant information) do not need to change, because they are also part of the set of information that is replicated. The administrator is responsible for creating and maintaining any backups.

Additionally, setting up SKLM key server clones can help gain some performance advantage by distributing MEK retrieval requests across the different clones in a round-robin fashion. To achieve this result, the administrator must specify different orderings of the server endpoints on different IBM Spectrum Scale nodes in the `/var/mmfs/etc/RKM.conf` file.

For example, if two cloned SKLM servers are available (such as `tls://keysrv.ibm.com:5696` and `tls://keysrv_backup.ibm.com:5696`), half of the nodes in the cluster can have the following content in `/var/mmfs/etc/RKM.conf`:

```
...
   kmipServerUri  = tls://keysrv.ibm.com:5696
   kmipServerUri2 = tls://keysrv_backup.ibm.com:5696
...
```

The other half can use the following content:

```
...
   kmipServerUri  = tls://keysrv_backup.ibm.com:5696
   kmipServerUri2 = tls://keysrv.ibm.com:5696
...
```

The client keystore directory and its files

The files in the client keystore directory include the client keystore file, the public and private key files for the client, and possibly other files that are described in later topics.

The management of these files depends on the setup:
* In the simplified setup, the **mmkeyserv** command creates and updates the files in the client keystore directory automatically.
* In the regular setup and the Vormetric setup, you must run various commands to create and update the files.

The location of the client keystore directory also depends on the setup:

Table 50. The client keystore directory

Setup	Location of the client keystore directory
Simplified setup	`/var/mmfs/ssl/keyServ`
Regular setup	`/var/mmfs/etc/RKMcerts`
Vormetric DSM setup	`/var/mmfs/etc/RKMcerts`

Establishing an encryption-enabled environment

Establishing an encryption-ready environment requires a sequence of activities which depend on the type and version of the RKM, as well as on the version of IBM Spectrum Scale.

Each summary covers a basic setup with a single encrypted fileset. There are currently three supported deployment scenarios:
* ISKLM version 2.5.0.4 or later (including 2.6) and IBM Spectrum Scale 4.2.1 or later
* Vormetric DSM version 5.2.3 or later and IBM Spectrum Scale 4.2.1 or later

- ISKLM version 2.5.0.1 or later and GPFS Advanced Edition V4.1 or later or IBM Spectrum Scale V4.2 or later

Simplified setup: Using SKLM with a self-signed certificate

Learn how to configure SKLM in the simplified setup when you use a self-signed server certificate rather than a certificate chain from a certificate authority.

This topic describes the simplified method for setting up encryption with IBM Security Key Lifecycle Manager (SKLM) as the key management server and with a self-signed certificate on the KMIP port of the SKLM server. If your deployment scenario uses a chain of certificates from a certificate authority, see one of the following topics:

The simplified setup with IBM Security Key Lifecycle Manager (SKLM) requires IBM Spectrum Scale Advanced Edition or IBM Spectrum Scale Data Management Edition V4.2.1 or later and SKLM V2.5.0.4 or later (including V2.6).

Note: If you are using SKLM v2.7 or later, see the topic "Configuring encryption with SKLM v2.7" on page 614.

Be aware of the following considerations:

- The IBM Spectrum Scale node that you are configuring for encryption must have direct network access to the system where the key server is installed.

  SECURITY NOTE: The contents of the following files are security-sensitive:
  - The `RKM.conf` file
    - For the simplified setup, the file is at `/var/mmfs/ssl/keyServ/RKM.conf`.
  - The directory for the client keystore
    - For the simplified setup, the directory is at `/var/mmfs/ssl/keyServ`.

  IBM Spectrum Scale reads the contents of security-sensitive files only if the following conditions are met:
  - They are regular files that are owned by the root user.
  - They are in the root group.
  - They are readable and writable only by the user.

  See the permission bits in the following example:
  ```
  -rw-------. 1 root root 2454 Mar 20 10:32 /var/mmfs/ssl/keyServ/RKM.conf
  drw-------. 2 root root 4096 Mar 20 11:15 /var/mmfs/ssl/keyServ/
  -rw-------. 1 root root 3988 Mar 20 11:15 /var/mmfs/ssl/keyServ/keystore_name.p12
  ```

  Note: In the simplified setup, the **mmkeyserv** command sets the permission bits automatically.
- **CAUTION:**
  It is a good practice to take the following precautions:
  - **Ensure that the passphrase for the client certificate file is not leaked through other means, such as the shell history.**
  - **Take appropriate precautions to ensure that the security-sensitive files are not lost or corrupted. IBM Spectrum Scale does not manage or replicate the files.**

  Important: The client keystore must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version

of NFS or store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

The setup is greatly simplified by the use of the **mmkeyserv** command, which can communicate with and configure the SKLM server from the IBM Spectrum Scale node. The **mmkeyserv** command automates the following tasks:
- Creating and configuring the client credentials of the IBM Spectrum Scale node.
- Creating a device group and master encryption keys for the node on SKLM.
- Creating an RKM stanza in the `RKM.conf` configuration file.
- Retrieving a server certificate from SKLM and storing it in the PKCS#12 keystore of the client.
- Propagating the encryption configuration and credentials to all the nodes in the IBM Spectrum Scale cluster.

See the following subtopics for instructions:
- "Part 1: Installing Security Key Lifecycle Manager"
- "Part 2: Configuring a node for encryption" on page 570

Part 1: Installing Security Key Lifecycle Manager

Follow the instructions in this subtopic to install and configure the IBM Security Key Lifecycle Manager (SKLM).

1. Install IBM Security Key Lifecycle Manager. For the required version, see "Preparation for encryption" on page 563. For the installation, choose a system that the IBM Spectrum Scale node that you want to configure has direct network access to. For information about installing SKLM, see the *Installing and configuring* chapter of the SKLM documentation.
2. From the main page of the SKLM web GUI, click **Configuration** > **Key Serving Parameters** and select the check box for **Keep pending client device communication certificates**.
3. Configure SKLM to have the same FIPS 140-2 (FIPS) setting as the IBM Spectrum Scale cluster. Follow these steps:
   a. Determine the FIPS setting of the cluster by entering the following command on the command line:

      ```
      mmlsconfig FIPS1402mode
      ```

      The command returns `yes` if the cluster complies with FIPS or `no` if not.
   b. On the SKLM server system, open the `SKLMConfig.properties` file.

      Note: The default location of the `SKLMConfig.properties` file depends on the operating system:
      - On AIX, Linux, and similar operating systems the directory is at the following location:

         ```
         /opt/IBM/WebSphere/AppServer/products/sklm/config/SKLMConfig.properties
         ```
      - On Microsoft Windows the directory is at the following location:

         ```
         Drive:\Program Files (x86)\IBM\WebSphere\AppServer\products\sklm\config\
         SKLMConfig.properties
         ```
   c. Add or remove the following line from the `SKLMConfig.properties` file. Add the line to configure SKLM to comply with FIPS, or remove it to have SKLM not comply with FIPS.

      ```
      fips=on
      ```
4. Configure the SKLM server to have the same NIST SP800-131a (NIST) setting as the IBM Spectrum Scale cluster. Follow these steps:
   a. Determine the NIST setting of the cluster by entering the following command on the command line:

      ```
      mmlsconfig nistCompliance
      ```

      The command returns `SP800-131A` if the cluster complies with NIST or `off` if not.

b. On the SKLM server system, open the `SKLMConfig.properties` file. For the location of this file, see the note in Step 3.

c. Add the following line to configure SKLM to comply with NIST or remove it to configure SKLM not to comply with NIST:

```
TransportListener.ssl.protocols=TLSv1.2
```

d. For all V2.5.0.x versions of SKLM, if you are configuring SKLM to comply with NIST, modify the following variable to include only cipher suites that are approved by NIST. The following statement is all on one line, with no space before or after the comma:

```
TransportListener.ssl.ciphersuites=TLS_RSA_WITH_AES_256_CBC_SHA256,
    TLS_RSA_WITH_AES_128_CBC_SHA256
```

5. Configure IBM WebSphere® Application Server so that it has the same NIST setting as the IBM Spectrum Scale cluster. See the topic Transitioning WebSphere Application Server to the SP800-131 security standard in the volume *WebSphere Application Server Network Deployment* in the WebSphere Application Server online documentation.

   - WebSphere Application Server can be configured to run SP800-131 in a `transition` mode or a `strict` mode. The `strict` mode is recommended.

   - When NIST is enabled, make sure that WebSphere Application Server certificate size is at least 2048 bytes and is signed with SHA256withRSA as described in the preceding link.

6. If the cipher suites were set at any time, SKLM 2.6.0.0 has a known issue that causes server certificates always to be signed with SHA1withRSA. To work around the problem, follow these steps:

   a. While the SKLM server is running, in the `SKLMConfig.properties` file, modify the `requireSHA2Signatures` property as follows:

```
requireSHA2Signatures=true
```

   b. Do not restart the server.

   c. Generate a new server certificate and set it to be the one in use.

   d. If you restart the server, you must repeat this workaround before you can create a server certificate that is signed other than with SHA1withRSA.

7. Create a self-signed server certificate:

   a. On the system where SKLM is running, open the graphical user interface.

   b. Click **Configuration** > **SSL/KMIP**.

   c. Click **Create self-signed certificate**.

   d. Enter the information for the certificate and click **OK**.

   e. Restart the server to verify that the server can operate with the new certificate.

Part 2: Configuring a node for encryption

Gather the following information:

- The logon password of the `SKLMAdmin` administrator of the SKLM server
- The certificate chain of the SKLM server (optional)

The following table provides an overview of the configuration process. The steps in the table correspond to the steps in the procedure that begins immediately after the table.

Table 51. Configuring a node for encryption in the simplified setup

Step number	Steps
Step 1	Verify the direct network connection between the IBM Spectrum Scale node and the SKLM server.
Step 2	Add the SKLM key server to the configuration.
Step 3	Add a tenant to the key server.
Step 4	Create a key client.

Table 51. Configuring a node for encryption in the simplified setup (continued)

Step number	Steps
Step 5	Register the key client to the tenant.
Step 6	Create a master encryption key in the tenant.
Step 7	Set up an encryption policy in the cluster.
Step 8	Test the encryption policy.

Follow these steps:

1. Verify that the IBM Spectrum Scale node that you are configuring for encryption has a direct network connection to the system on which the SKLM key server runs.

2. Use the **mmkeyserv server add** command to add the SKLM key server from Part I to the configuration:

   a. Issue the following command:

   ```
   mmkeyserv server add ServerName
   ```

   where *ServerName* is the host name or IP address of the SKLM key server that you want to add. See the example listing in Figure 17 on page 572. You can also specify the REST port number of the SKLM key server:

   ```
   mmkeyserv server add ServerName --port RestPortNumber
   ```

   The default REST port number is 443 for SKLM v2.7 and later and 9080 for SKLM v2.6 and earlier.

   b. Enter the password for the SKLM server when prompted.

   c. To view the certificate chain of the SKLM server, enter `view` when prompted.

   d. Verify that the certificates that are displayed have the same contents as the certificates in the chain that you downloaded from SKLM.

   e. Enter `yes` to trust the certificates or `no` to reject them. If you trust the certificates, the command adds the key server object to the configuration. In the following listing, key server `keyserver01` is added:

```
# mmkeyserv server add keyserver01
Enter password for the key server keyserver01:
The security certificate(s) from keyserver01.gpfs.net must be accepted to continue.  View the
certificate(s) to determine whether you want to trust the certifying authority.
Do you want to view or trust the certificate(s)? (view/yes/no) view

Serial number:          01022a8adf20f3
SHA-256 digest:         2ca4a48a3038f37d430162be8827d91eb584e98f5b3809047ef4a1c72e15fc4c
Signature:              7f0312e7be18efd72c9d8f37dbbb832724859ba4bb5827c230e2161473e0753b367ed49d
993505bd23858541475de8e021e0930725abbd3d25b71edc8fc3de20b7c2db5cd4e865f41c7c410c1d710acf222e1c4
5189108e40568ddcbeb21094264da60a1d96711015a7951eb2655363309d790ab44ee7b26adf8385e2c210b8268c5ae
de5f82f268554a6fc22ece6efeee2a6264706e71416a0dbe8c39ceacd86054d7cc34dda4fffea4605c037d321290556
10821af85dd9819a4d7e4baa70c51addcda720d33bc9f8bbde6d292c028b2f525a0275ebea968c26f8f0c4b604719ae
3b04e71ed7a8188cd6adf68764374b29c91df3d101a941bf8b7189485ad72
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, CN=c40bbc1xn3.gpfs.net

Serial number:          01022a24475466
SHA-256 digest:         077c3b53c5046aa893b760c11cca3a993efbc729479771e03791f9ed4f716879
Signature:              227b5befe89f2e55ef628da6b50db1ab842095a54e1505655e3d95fee753a7f7554868a
a79b294c503dc34562cf69c2a20128796758838968565c0812c4aedbb0543d396646a269c02bf4c5ce5acba4409a10e
ffbd47ca38ce492698e2dcdc8390b9ae3f4a47c23ee3045ff0145218668f35a63edac68201789ed0db6e5c170f5c6db
49769f0b4c9a5f208746e4342294c447793ed087fa0ac762588faf420febeb3fca411e4e725bd46476e1f9f44759a69
6573af5dbbc9553218c7083c80440f2e542bf56cc5cc18156cce05efd6c2e5fea2b886c5c1e262c10af18b13ccf38c3
533ba025b97bbe62f271545b2ab5c1f50c1dca45ce504dfcfc257362e9b43
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net

Do you trust the certificate(s) above? (yes/no) yes
```

Figure 17. Example listing for `mmkeyserv server add`

    f. Issue the **mmkeyserv server show** command to verify that the key server is added. The following listing shows that keyserver01 is created:

```
# mmkeyserv server show
keyserver01
        Type:               ISKLM
        Hostname:           keyserver01.gpfs.net
        User ID:            SKLMAdmin
        REST port:          9080
        Label:              1_keyserver01
        NIST:               on
        FIPS1402:           off
        Backup Key Servers:
        Distribute:         yes
        Retrieval Timeout:  120
        Retrieval Retry:    3
        Retrieval Interval: 10000
```

3. Issue the **mmkeyserv tenant add** command to add a tenant to the key server. The command creates the tenant on the SKLM server if it does not exist. A *tenant* is an entity on the SKLM server that can contain encryption keys and certificates. SKLM uses the term *device group* instead of *tenant*.

    a. Issue the following command to add tenant devG1 to key server keyserver01. Enter the password for the SKLM server when prompted:

```
# mmkeyserv tenant add devG1 --server keyserver01
Enter password for the key server keyserver01:
```

    b. Issue the **mmkeyserv tenant show** command to verify that the tenant is added. The following listing shows that tenant devG1 is added to keyserver01:

```
# mmkeyserv tenant show
devG1
    Key Server:       keyserver01.gpfs.net
    Registered Client:  (none)
```

4. Issue the **mmkeyserv client create** command to create a key client. A key client can request master encryption keys from a tenant after it is registered to the tenant. The command creates a client keystore on the IBM Spectrum Scale node and puts into it a set of client credentials and the certificate chain of the SKLM server. The directory of the keystore is:

`/var/mmfs/ssl/keyServ`

a. Issue the following command to create key client c1Client1 for key server keyserver01. Enter the password for the SKLM server and a passphrase for the new keystore when prompted:

```
# mmkeyserv client create c1Client1 --server keyserver01
Enter password for the key server keyserver01:
Create a pass phrase for keystore:
Confirm your pass phrase:
```

b. Issue the **mmkeyserv client show** command to verify that the key client is created. The following listing shows that key client c1Client1 is created for remote server keyserver01.gpfs.net:

```
# mmkeyserv client show
c1Client1
        Label:            c1Client1
        Key Server:       keyserver01.gpfs.net
        Tenants:          (none)
```

5. Issue the **mmkeyserv client register** command to register the key client with the tenant:

You must provide a remote key management (RKM) ID as an input for this command. An RKM ID identifies an RKM stanza, which is a structure on the node in which configuration information about the key client, the tenant, and the key server is stored. Each stanza holds the information about one connection between a key client, a tenant, and a key server.

It is a good practice to use a format like the following one to ensure that the RKM ID is unique:

keyServerName_tenantName

For example, the RKM ID for the key server and the tenant in these instructions is keyserver01_devG1.

a. Issue the following command to register key client c1Client1with tenant devG1 under RKM ID keyserver01_devG1. Enter the requested information when prompted:

```
# mmkeyserv client register c1Client1 --tenant devG1 --rkm-id keyserver01_devG1
Enter password for the key server:

mmkeyserv: [I] Client currently does not have access to the key.  Continue the registration
    process ...
mmkeyserv: Successfully accepted client certificate
```

b. Issue the command **mmkeyserv tenant show** to verify that the key client is known to the tenant. The following listing shows that tenant devG1 lists c1Client1 as a registered client:

```
mmkeyserv tenant show
devG1
    Key Server:       keyserver01.gpfs.net
    Registered Client:  c1Client1
```

c. You can also issue the command **mmkeyserv client show** to verify that the tenant is known to the client. The following listing shows that client c1Client1 is registered with tenant devG1:

```
# mmkeyserv client show
c1Client1
        Label:            c1Client1
        Key Server:       keyserver01.gpfs.net
        Tenants:          devG1
```

d. To see the contents of the RKM stanza, issue the **mmkeyserv rkm show** command. In the following listing, notice that the RKM ID of the stanza is keyserver01_devG1, the string that was specified in Step 5(a):

```
# mmkeyserv rkm show
keyserver01_devG1 {
type = ISKLM
kmipServerUri = tls://192.168.40.59:5696
keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
passphrase = pw4c1Client1
clientCertLabel = c1Client1
tenantName = devG1
}
```

e. You can also see the RKM stanza by displaying the contents of the RKM.conf file on the node:

```
# cat /var/mmfs/ssl/keyServ/RKM.conf
keyserver01_devG1 {
  type = ISKLM
  kmipServerUri = tls://192.168.40.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
  passphrase = pw4c1Client1
  clientCertLabel = c1Client1
  tenantName = devG1
}
```

6. Issue the **mmkeyserv key create** command to create a master encryption key in the tenant. The following command creates a master encryption key in tenant devG1 of server keyserver01.gpfs.net. The command displays the UUID of the encryption key (not the key value itself) at line 3 of the listing:

```
# mmkeyserv key create --server keyserver01.gpfs.net --tenant devG1
Enter password for the key server keyserver01.gpfs.net:
KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1
```

7. Set up an encryption policy on the node.

a. Create a policy that instructs GPFS to do the encryption tasks that you want. The following policy is an example policy that instructs IBM Spectrum Scale to encrypt all files in the file system with a file encryption key and to wrap the file encryption key with a master encryption key:

```
RULE 'p1' SET POOL 'system' # one placement rule is required at all times
RULE 'Encrypt all files in file system with rule E1'
SET ENCRYPTION 'E1'
WHERE NAME LIKE '%'
RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1:keyserver01_devG1')
```

In the last line of the policy, the character string within single quotation marks (') is the key name. A *key name* is a compound of two parts in the following format:

KeyID:RkmID

where:

KeyID Specifies the UUID of the key that you created in Step 6.

RkmID Specifies the RKM ID that you specified in Step 5(a).

b. Issue the **mmchpolicy** command to install the rule.

CAUTION:
Installing a new policy with the mmchpolicy command removes all the statements in the previous policy. To add statements to an existing policy without deleting the previous contents, collect all policy statements for the file system into one file. Add the new statements to the file and install the contents of the file with mmchpolicy.

1) Issue the following command to install the policy rules in file enc.pol for file system c1FileSystem1:

```
# mmchpolicy c1FileSystem1 /tmp/enc.pol
Validated policy `enc.pol': Parsed 3 policy rules.
Policy `enc.pol' installed and broadcast to all nodes.
```

2) You can list the new encryption policy with the following command:

```
        # mmlspolicy c1FileSystem1 -L
```

8. Test the new encryption policy

   a. Create a file in the file system c1FileSystem1:

      ```
      # echo 'Hello World!' >/c1FilesSystem1/hw.enc
      ```

      The policy engine detects the new file, encrypts it, and wraps the file encryption key in a master encryption key.

   b. To verify that the file hw.enc is encrypted, issue the following command to display the encryption attribute of the file. The output shows that the file is encrypted:

      ```
      # mmlsattr -n gpfs.Encryption /c1Filesystem1/hw.enc
      file name:            /c1Filesystem1/hw.enc
      gpfs.Encryption:      "EAGC????.????????????? ??????h????????????????? ?u?~?}????????????t??lN??
      'k???*?3??C???#?)?KEY-ef07b465-cfa5-4476-9f63-544e4b3cc119?NewGlobal11?"
      EncPar 'AES:256:XTS:FEK:HMACSHA512'
          type: wrapped FEK  WrpPar 'AES:KWRAP'  CmbPar 'XORHMACSHA512'
              KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1:keyserver01_devG1
      ```

Simplified setup: Using SKLM with a certificate chain

Learn how to configure SKLM in the simplified setup when you use a certificate chain from a certificate authority rather than a self-signed server certificate.

This topic describes the simplified method for setting up encryption with IBM Security Key Lifecycle Manager (SKLM) as the key management server and with a certificate signed by a certificate authority (CA) on the KMIP port of the SKLM server. If your deployment scenario uses a self-signed server certificate, see one of the following topics:

The simplified setup with IBM Security Key Lifecycle Manager (SKLM) requires IBM Spectrum Scale Advanced Edition or IBM Spectrum Scale Data Management Edition V4.1 or later and SKLM V2.5.0.1 or later (including V2.6).

Note: If you are using SKLM v2.7 or later, see the topic

Be aware of the following considerations:
- The IBM Spectrum Scale node that you are configuring for encryption must have direct network access to the system where the key server is installed.
-

SECURITY NOTE: The contents of the following files are security-sensitive:
- The RKM.conf file
   - For the simplified setup, the file is at /var/mmfs/ssl/keyServ/RKM.conf.
- The directory for the client keystore
   - For the simplified setup, the directory is at /var/mmfs/ssl/keyServ.

IBM Spectrum Scale reads the contents of security-sensitive files only if the following conditions are met:
- They are regular files that are owned by the root user.
- They are in the root group.
- They are readable and writable only by the user.

See the permission bits in the following example:

```
-rw-------. 1 root root 2454 Mar 20 10:32 /var/mmfs/ssl/keyServ/RKM.conf
drw-------. 2 root root 4096 Mar 20 11:15 /var/mmfs/ssl/keyServ/
-rw-------. 1 root root 3988 Mar 20 11:15 /var/mmfs/ssl/keyServ/keystore_name.p12
```

Note: In the simplified setup, the **mmkeyserv** command sets the permission bits automatically.

- **CAUTION:**
  It is a good practice to take the following precautions:
  - **Ensure that the passphrase for the client certificate file is not leaked through other means, such as the shell history.**
  - **Take appropriate precautions to ensure that the security-sensitive files are not lost or corrupted. IBM Spectrum Scale does not manage or replicate the files.**

-

  Important: The client keystore must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version of NFS or store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

The setup is greatly simplified by the use of the **mmkeyserv** command, which can communicate with and configure the SKLM server from the IBM Spectrum Scale node. The **mmkeyserv** command automates the following tasks:

- Creating and configuring the client credentials of the IBM Spectrum Scale node.
- Creating a device group and master encryption keys for the node on SKLM.
- Creating an RKM stanza in the `RKM.conf` configuration file.
- Retrieving a server certificate from SKLM and storing it in the PKCS#12 keystore of the client.
- Propagating the encryption configuration and credentials to all the nodes in the IBM Spectrum Scale cluster.

See the following subtopics for instructions:

Part 1: Installing Security Key Lifecycle Manager

Follow the instructions in this subtopic to install and configure the IBM Security Key Lifecycle Manager (SKLM).

1. Install IBM Security Key Lifecycle Manager. For the required version, see "Preparation for encryption" on page 563. For the installation, choose a system that the IBM Spectrum Scale node that you want to configure has direct network access to. For information about installing SKLM, see the *Installing and configuring* chapter of the SKLM documentation.

2. From the main page of the SKLM web GUI, click **Configuration** > **Key Serving Parameters** and select the check box for **Keep pending client device communication certificates**.

3. Configure SKLM to have the same FIPS 140-2 (FIPS) setting as the IBM Spectrum Scale cluster. Follow these steps:

   a. Determine the FIPS setting of the cluster by entering the following command on the command line:

   ```
   mmlsconfig FIPS1402mode
   ```

   The command returns `yes` if the cluster complies with FIPS or `no` if not.

   b. On the SKLM server system, open the `SKLMConfig.properties` file.

Note: The default location of the `SKLMConfig.properties` file depends on the operating system:

- On AIX, Linux, and similar operating systems the directory is at the following location:

    `/opt/IBM/WebSphere/AppServer/products/sklm/config/SKLMConfig.properties`

- On Microsoft Windows the directory is at the following location:

    `Drive:\Program Files (x86)\IBM\WebSphere\AppServer\products\sklm\config\`
    `SKLMConfig.properties`

   c. Add or remove the following line from the `SKLMConfig.properties` file. Add the line to configure SKLM to comply with FIPS, or remove it to have SKLM not comply with FIPS.

   `fips=on`

4. Configure the SKLM server to have the same NIST SP800-131a (NIST) setting as the IBM Spectrum Scale cluster. Follow these steps:

   a. Determine the NIST setting of the cluster by entering the following command on the command line:

   `mmlsconfig nistCompliance`

   The command returns `SP800-131A` if the cluster complies with NIST or `off` if not.

   b. On the SKLM server system, open the `SKLMConfig.properties` file. For the location of this file, see the note in Step 3.

   c. Add the following line to configure SKLM to comply with NIST or remove it to configure SKLM not to comply with NIST:

   `TransportListener.ssl.protocols=TLSv1.2`

   d. For all V2.5.0.x versions of SKLM, if you are configuring SKLM to comply with NIST, modify the following variable to include only cipher suites that are approved by NIST. The following statement is all on one line, with no space before or after the comma:

   `TransportListener.ssl.ciphersuites=TLS_RSA_WITH_AES_256_CBC_SHA256,`
   `    TLS_RSA_WITH_AES_128_CBC_SHA256`

5. Configure IBM WebSphere Application Server so that it has the same NIST setting as the IBM Spectrum Scale cluster. See the topic Transitioning WebSphere Application Server to the SP800-131 security standard in the volume *WebSphere Application Server Network Deployment* in the WebSphere Application Server online documentation.

   - WebSphere Application Server can be configured to run SP800-131 in a `transition` mode or a `strict` mode. The `strict` mode is recommended.

   - When NIST is enabled, make sure that WebSphere Application Server certificate size is at least 2048 bytes and is signed with SHA256withRSA as described in the preceding link.

6. If the cipher suites were set at any time, SKLM 2.6.0.0 has a known issue that causes server certificates always to be signed with SHA1withRSA. To work around the problem, follow these steps:

   a. While the SKLM server is running, in the `SKLMConfig.properties` file, modify the `requireSHA2Signatures` property as follows:

   `requireSHA2Signatures=true`

   b. Do not restart the server.

   c. Generate a new server certificate signing request (CSR) to a third-party certificate authority (CA) and send it to the CA.

   d. When you receive the certificate from the third-party CA, import it into SKLM and set it to be the certificate in use. For more information, see the next subtopic.

   e. If you restart the server, you must repeat this workaround before you can create a server certificate that is signed other than with SHA1withRSA.

Part 2: Configuring SKLM

To configure SKLM, you must create a certificate signing request (CSR), send it to the certificate authority (CA), obtain the certificate chain from the CA, and import the root certificate into the SKLM server.

Note: For more information about the steps in this subtopic, see the steps that are described in the SKLM documentation, in the topic "Scenario: Request for a third-party certificate" at http://www.ibm.com/support/knowledgecenter/en/SSWPVP_2.7.0/com.ibm.sklm.doc/scenarios/cpt/cpt_ic_scenar_ca_certusage.html.

1. Create a certificate signing request (CSR) with the SKLM command-line interface:

   a. On the SKLM server system, open a command-line window.

   b. Change to the *WAS_HOME*/bindirectory. The location of this directory depends on the operating system:

      - On AIX, Linux, and similar operating systems the directory is at the following location:

        ```
        /opt/IBM/WebSphere/AppServer/bin
        ```

      - On Microsoft Windows the directory is at the following location:

        ```
        drive:\Program Files (x86)\IBM\WebSphere\AppServer\bin
        ```

   c. Start the command-line interface to SKLM:

      - On AIX, Linux, and similar operating systems, enter the following command:

        ```
        ./wsadmin.sh -username SKLMAdmin -password mypwd -lang jython
        ```

      - On Microsoft Windows, enter the following command:

        ```
        wsadmin -username SKLMAdmin -password mypwd -lang jython
        ```

   d. In the SKLM command-line interface, enter the following command on one line:

      ```
      print AdminTask.tklmCertGenRequest('[-alias labelCsr -cn server
      -validity daysValid -keyStoreName defaultKeyStore -fileName fileName -usage SSLSERVER]')
      ```

      where:

      -alias labelCsr
      　　Specifies the certificate label of the CSR.

      -cn server
      　　Specifies the common name of the server in the certificate.

      -validity daysValid
      　　Specifies the validity period of the certificate in days.

      -keyStoreName defaultKeyStore
      　　Specifies the keystore name within SKLM where the CSR is stored. Typically, you should specify defaultKeyStore as the name here.

      -fileName fileName
      　　Specifies the fully qualified path of the directory where the CSR is stored on the SKLM server system, for example /root/sklmServer.csr.

      -usage SSLSERVER
      　　Specifies how the generated certificate is used in SKLM.

      SKLM responds with output like the following example:

      ```
      CTGKM0001I Command succeeded
      fileName
      ```

2. Send the CSR file from Step 1 to the certificate authority.

3. When you receive the generated certificate file, or *endpoint certificate* file, from the certificate authority, copy it to a directory on the node that you are configuring for encryption. For example, you might copy it to the directory and file /opt/IBM/WebSphere/AppServer/products/sklm/data/sklmServer.cert.

   Important: You must also obtain and copy the intermediate certificate files of the certificate chain of authority into the same temporary directory. The intermediate certificates might be included with the generated certificate file, or you might have to obtain the intermediate certificates separately.

Whatever the method, you must have a separate certificate file for the root certificate and for each intermediate certificate in the chain of authority. You need these certificate files in Part 3.

4. Import the root certificate into the SKLM server with the SKLM graphical user interface:

   a. On the **Welcome** page, in the **Action Items** section, in the **Key Groups and Certificates** area, click **You have pending certificates**.

   b. In the **Pending Certificates** table, click the certificate that you want to import and click**Import**.

   c. In the **File name and location** field, type the path and file name of the certificate file and click **Import**.

Part 3: Configuring a node for encryption

Gather the following information:

- The logon password of the SKLMAdmin administrator of the SKLM server
- The certificate chain of the SKLM server (optional)

The following table provides a high-level overview of the configuration process. The steps in the table correspond to the steps in the procedure that begins immediately after the table.

Table 52. Configuring a node for encryption in the simplified setup

Step number	Steps
Step 1	Verify the direct network connection between the IBM Spectrum Scale node and the SKLM server.
Step 2	Add the SKLM key server to the configuration.
Step 3	Add a tenant to the key server.
Step 4	Create a key client.
Step 5	Register the key client to the tenant.
Step 6	Create a master encryption key in the tenant.
Step 7	Set up an encryption policy in the cluster.
Step 8	Test the encryption policy.

1. Verify that the IBM Spectrum Scale node that you are configuring for encryption has a direct network connection to the system on which the SKLM key server runs.

2. Use the **mmkeyserv server add** command to add the SKLM key server from Part I to the configuration:

   a. Add the key server to the current configuration:

      1) Copy the files in the server certificate chain into a directory on the node that you are configuring. A good location is the same directory in which the keystore.pwd file is located.

      2) Rename each file with the same file name prefix, followed by an increasing integer value that indicates the order of the certificate in the chain, followed by the suffix .cert. Start the numbering with 0 for the root certificate.

         The following example shows the renamed certificate files for a server certificate chain. The chain consists of a root CA certificate, one intermediate certificate, and an endpoint certificate. The files are in directory /root and the file name prefix is sklmChain:

         /root/sklmChain0.cert (Root certificate)

         /root/sklmChain1.cert (Intermediate certificate)

         /root/sklmChain2.cert (Endpoint certificate)

         Certificate chains can contain more than one intermediate certificate.

      3) Issue the following command to add the key server to the current configuration:

         mmkeyserv server add *ServerName* --kmip-cert *CertFilesPrefix*

where:

ServerName
> Is the host name or IP address of the SKLM key server that you want to add.

CertFilesPrefix
> Specifies the path and the file name prefix of the files in the certificate chain. For the files from the example in the previous step, the path and file name prefix is `/root/sklmChain`.
>
> For more information, see the topic *mmkeyserv command* in the *IBM Spectrum Scale: Command and Programming Reference*.

b. Issue the following command:

```
mmkeyserv server add ServerName
```

where *ServerName* is the host name or IP address of the SKLM key server that you want to add. See the example listing in Figure 18 on page 581. You can also specify the REST port number of the SKLM key server:

```
mmkeyserv server add ServerName --port RestPortNumber
```

The default REST port number is 443 for SKLM v2.7 and later and 9080 for SKLM v2.6 and earlier.

c. Enter the password for the SKLM server when prompted.

d. To view the certificate chain of the SKLM server, enter `view` when prompted.

e. Verify that the certificates that are displayed have the same contents as the certificates in the chain that you downloaded from SKLM.

f. Enter `yes` to trust the certificates or `no` to reject them. If you trust the certificates, the command adds the key server object to the configuration. In the following listing, key server `keyserver01` is added:

```
# mmkeyserv server add keyserver01
Enter password for the key server keyserver01:
The security certificate(s) from keyserver01.gpfs.net must be accepted to continue.  View the
certificate(s) to determine whether you want to trust the certifying authority.
Do you want to view or trust the certificate(s)? (view/yes/no) view

Serial number:          01022a8adf20f3
SHA-256 digest:         2ca4a48a3038f37d430162be8827d91eb584e98f5b3809047ef4a1c72e15fc4c
Signature:              7f0312e7be18efd72c9d8f37dbb832724859ba4bb5827c230e2161473e0753b367ed49d
993505bd23858541475de8e021e0930725abbd3d25b71edc8fc3de20b7c2db5cd4e865f41c7c410c1d710acf222e1c4
5189108e40568ddcbeb21094264da60a1d96711015a7951eb2655363309d790ab44ee7b26adf8385e2c210b8268c5ae
de5f82f268554a6fc22ece6efeee2a6264706e71416a0dbe8c39ceacd86054d7cc34dda4fffea4605c037d321290556
10821af85dd9819a4d7e4baa70c51addcda720d33bc9f8bbde6d292c028b2f525a0275ebea968c26f8f0c4b604719ae
3b04e71ed7a8188cd6adf68764374b29c91df3d101a941bf8b7189485ad72
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, CN=c40bbc1xn3.gpfs.net

Serial number:          01022a24475466
SHA-256 digest:         077c3b53c5046aa893b760c11cca3a993efbc729479771e03791f9ed4f716879
Signature:              227b5befe89f2e55ef628da6b50db1ab842095a54e1505655e3d95fee753a7f7554868a
a79b294c503dc34562cf69c2a20128796758838968565c0812c4aedbb0543d396646a269c02bf4c5ce5acba4409a10e
ffbd47ca38ce492698e2dcdc8390b9ae3f4a47c23ee3045ff0145218668f35a63edac68201789ed0db6e5c170f5c6db
49769f0b4c9a5f208746e4342294c447793ed087fa0ac762588faf420febeb3fca411e4e725bd46476e1f9f44759a69
6573af5dbbc9553218c7083c80440f2e542bf56cc5cc18156cce05efd6c2e5fea2b886c5c1e262c10af18b13ccf38c3
533ba025b97bbe62f271545b2ab5c1f50c1dca45ce504dfcfc257362e9b43
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net

Do you trust the certificate(s) above? (yes/no) yes
```

Figure 18. Example listing for `mmkeyserv server add`

g. Issue the **mmkeyserv server show** command to verify that the key server is added. The following listing shows that keyserver01 is created:

```
# mmkeyserv server show
keyserver01
        Type:             ISKLM
        Hostname:         keyserver01.gpfs.net
        User ID:          SKLMAdmin
        REST port:        9080
        Label:            1_keyserver01
        NIST:             on
        FIPS1402:         off
        Backup Key Servers:
        Distribute:       yes
        Retrieval Timeout: 120
        Retrieval Retry:  3
        Retrieval Interval: 10000
```

3. Issue the **mmkeyserv tenant add** command to add a tenant to the key server. The command creates the tenant on the SKLM server if it does not exist. A *tenant* is an entity on the SKLM server that can contain encryption keys and certificates. SKLM uses the term *device group* instead of *tenant*.

a. Issue the following command to add tenant devG1 to key server keyserver01. Enter the password for the SKLM server when prompted:

```
# mmkeyserv tenant add devG1 --server keyserver01
Enter password for the key server keyserver01:
```

b. Issue the **mmkeyserv tenant show** command to verify that the tenant is added. The following listing shows that tenant devG1 is added to keyserver01:

```
# mmkeyserv tenant show
devG1
    Key Server:       keyserver01.gpfs.net
    Registered Client:  (none)
```

4. Issue the **mmkeyserv client create** command to create a key client. A key client can request master encryption keys from a tenant after it is registered to the tenant. The command creates a client keystore on the IBM Spectrum Scale node and puts into it a set of client credentials and the certificate chain of the SKLM server. The directory of the keystore is:

`/var/mmfs/ssl/keyServ`

   a. Issue the following command to create key client `c1Client1` for key server `keyserver01`. Enter the password for the SKLM server and a passphrase for the new keystore when prompted:

   ```
   # mmkeyserv client create c1Client1 --server keyserver01
   Enter password for the key server keyserver01:
   Create a pass phrase for keystore:
   Confirm your pass phrase:
   ```

   b. Issue the **mmkeyserv client show** command to verify that the key client is created. The following listing shows that key client `c1Client1` is created for remote server keyserver01.gpfs.net:

   ```
   # mmkeyserv client show
   c1Client1
         Label:            c1Client1
         Key Server:       keyserver01.gpfs.net
         Tenants:          (none)
   ```

5. Issue the **mmkeyserv client register** command to register the key client with the tenant:

   You must provide a remote key management (RKM) ID as an input for this command. An RKM ID identifies an RKM stanza, which is a structure on the node in which configuration information about the key client, the tenant, and the key server is stored. Each stanza holds the information about one connection between a key client, a tenant, and a key server.

   It is a good practice to use a format like the following one to ensure that the RKM ID is unique:

   keyServerName_tenantName

   For example, the RKM ID for the key server and the tenant in these instructions is `keyserver01_devG1`.

   a. Issue the following command to register key client `c1Client1`with tenant devG1 under RKM ID `keyserver01_devG1`. Enter the requested information when prompted:

   ```
   # mmkeyserv client register c1Client1 --tenant devG1 --rkm-id keyserver01_devG1
   Enter password for the key server:

   mmkeyserv: [I] Client currently does not have access to the key.  Continue the registration
       process ...
   mmkeyserv: Successfully accepted client certificate
   ```

   b. Issue the command **mmkeyserv tenant show** to verify that the key client is known to the tenant. The following listing shows that tenant devG1 lists `c1Client1` as a registered client:

   ```
   mmkeyserv tenant show
   devG1
       Key Server:       keyserver01.gpfs.net
       Registered Client:  c1Client1
   ```

   c. You can also issue the command **mmkeyserv client show** to verify that the tenant is known to the client. The following listing shows that client `c1Client1` is registered with tenant devG1:

   ```
   # mmkeyserv client show
   c1Client1
         Label:            c1Client1
         Key Server:       keyserver01.gpfs.net
         Tenants:          devG1
   ```

   d. To see the contents of the RKM stanza, issue the **mmkeyserv rkm show** command. In the following listing, notice that the RKM ID of the stanza is `keyserver01_devG1`, the string that was specified in Step 5(a):

```
# mmkeyserv rkm show
keyserver01_devG1 {
type = ISKLM
kmipServerUri = tls://192.168.40.59:5696
keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
passphrase = pw4c1Client1
clientCertLabel = c1Client1
tenantName = devG1
}
```

e. You can also see the RKM stanza by displaying the contents of the RKM.conf file on the node:

```
# cat /var/mmfs/ssl/keyServ/RKM.conf
keyserver01_devG1 {
  type = ISKLM
  kmipServerUri = tls://192.168.40.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
  passphrase = pw4c1Client1
  clientCertLabel = c1Client1
  tenantName = devG1
}
```

6. Issue the **mmkeyserv key create** command to create a master encryption key in the tenant. The following command creates a master encryption key in tenant devG1 of server keyserver01.gpfs.net. The command displays the UUID of the encryption key (not the key value itself) at line 3 of the listing:

```
# mmkeyserv key create --server keyserver01.gpfs.net --tenant devG1
Enter password for the key server keyserver01.gpfs.net:
KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1
```

7. Set up an encryption policy on the node.

   a. Create a file management policy that instructs GPFS to do the encryption tasks that you want. The following is an example policy. It instructs IBM Spectrum Scale to encrypt all files in the file system with a file encryption key (FEK) and to wrap the FEK with a master encryption key (MEK):

```
RULE 'p1' SET POOL 'system' # one placement rule is required at all times
RULE 'Encrypt all files in file system with rule E1'
SET ENCRYPTION 'E1'
WHERE NAME LIKE '%'
RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1:keyserver01_devG1')
```

   In the last line of the policy, the character string within single quotation marks (') is the key name. A *key name* is a compound of two parts in the following format:

   KeyID:RkmID

   where:

   KeyID   Specifies the UUID of the key that you created in Step 6.

   RkmID   Specifies the RKM ID that you specified in Step 5(a).

   b. Issue the **mmchpolicy** command to install the rule.

   CAUTION:
   Installing a new policy with the mmchpolicy command removes all the statements in the previous policy. To add statements to an existing policy without deleting the previous contents, collect all policy statements for the file system into one file. Add the new statements to the file and install the contents of the file with the mmchpolicy command.

   1) Issue the following command to install the policy rules in file enc.pol for file system c1FileSystem1:

```
# mmchpolicy c1FileSystem1 /tmp/enc.pol
Validated policy `enc.pol': Parsed 3 policy rules.
Policy `enc.pol' installed and broadcast to all nodes.
```

| 2) You can list the new encryption policy with the following command:

| `# mmlspolicy c1FileSystem1 -L`

| 8. Test the new encryption policy

| a. Create a file in the file system `c1FileSystem1`:

| `# echo 'Hello World!' >/c1FilesSystem1/hw.enc`

| The policy engine detects the new file, encrypts it, and wraps the file encryption key in a master
| encryption key.

| b. To verify that the file `hw.enc` is encrypted, issue the following command to display the encryption
| attribute of the file. The output shows that the file is encrypted:

```
# mmlsattr -n gpfs.Encryption /c1Filesystem1/hw.enc
file name:           /c1Filesystem1/hw.enc
gpfs.Encryption:     "EAGC????.????????????? ??????h????????????????? ?u?~?}????????????t??lN??
'k???*?3??C???#?)?KEY-ef07b465-cfa5-4476-9f63-544e4b3cc119?NewGlobal11?"
EncPar 'AES:256:XTS:FEK:HMACSHA512'
     type: wrapped FEK  WrpPar 'AES:KWRAP'  CmbPar 'XORHMACSHA512'
         KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1:keyserver01_devG1
```

Simplified setup: Valid and invalid configurations

Considerable flexibility and a few restrictions govern the registering of key clients with tenants.

Single cluster, single key server

With a single cluster and a single key server, the following rules apply:

- A single key client can register with more than one tenant.
- However, two or more key clients cannot register with the same tenant.

The following figure illustrates these rules:

- Key client `c1Client1` can register with tenants `devG1`, `devG2`, and `devG3`.
- But key client `c1Client2` cannot register with `devG1` (or `devG2` or `devG3`) because `c1Client1` is already registered there.
- Tenant `devG4` is added so that key client `c1Client2` can register with a tenant.

Figure 19. Single cluster, single key server

Multiple clusters, single key server

With multiple clusters and a single key server, more than one key client can register with a tenant if the key clients are in different clusters.

The following figure illustrates these rules:

- With key clients `c1Client1` in `Cluster1` and `c2Client1` in `Cluster2`:
  - `c1Client1` is registered with tenants `devG1`, `devG2`, and `devG3`.
  - `c2Client1` can also register with `devG1`, `devG2`, and `devG3`, because it is in a different cluster.
- Similarly, with `c1Client2` in `Cluster1` and `c2Client1` in `Cluster2`:
  - `c1Client2` is registered with tenant `devG4`.
  - `c2Client1` can also register with `devG4`, because `c2Client1` is in a different cluster.

Figure 20. Multiple clusters, single key server

Single cluster, multiple key servers

With a single cluster and multiple key servers, the following rules apply:
- Different key clients in the same cluster can register with different tenants in the same key server.
- But a single key client cannot register with tenants in different key servers.

The following figure illustrates these rules:
- With key clients `c1Client1` and `c1Client2`, both in `Cluster1`, it is the same situation as in Figure 19 on page 585.
  - `c1Client1` is registered with tenants `devG1`, `devG2`, and `devG3` in `keyserver01`.
  - `c1Client2` can register with tenant `devG4` in (but not with `devG1`, `devG2`, or `devG3`).
- With key client `c1Client2` in `Cluster1`:
  - `c1Client2` can register with a tenant (`devG4` in this example) in .
  - But `c1Client2` cannot also register with a tenant (`devG3`) in `keyserver02`.
- `c1Client3` was created in `Cluster1` to register with tenants `devG1` and `devG2` in `keyserver02`.

Figure 21. Single cluster, multiple key servers

Simplified setup: Accessing a remote file system

See an example of how to access an encrypted file in a remote cluster.

This topic shows how to configure a cluster so that it can mount an encrypted file system that is in another cluster. In the examples in this topic, the encrypted file system is `c1FileSystem1` and its cluster is `Cluster1`. The cluster that mounts the encrypted file system is `Cluster2`.

The examples assume that `Cluster1` and `c1FileSystem1` are the cluster and file system that you configured in the topic "Simplified setup: Using SKLM with a self-signed certificate" on page 568. You configured `Cluster1` for encryption and you created a policy that caused all the files in `c1FileSystem1` be encrypted.

To configure `Cluster2` with remote access to an encrypted file in `Cluster1`, you must configure `Cluster2` for encryption in much the same way that `Cluster1` was configured. As the following table shows, `Cluster2` must add the same key server and tenant as `Cluster1`. However, `Cluster2` must create its own key client and register it with the tenant.

Note: In the third column of the table, items in square brackets are connected or added during this topic. The fourth column shows the step in which each item in the third column is added.

Table 53. Setup of Cluster1 and Cluster2

Item	Cluster1	Cluster2	Steps
File system	`c1FileSystem1`	`[c1FileSystem1_Remote]`	Step 1
Connected to a key server	`keyserver01`	`[keyserver01]`	Step 2
Connected to a tenant	`c1Tenant1 on keyserver01`	`[c1Tenant1 on keyserver01]`	Step 3
Created a key client	`c1Client1`	`[c2Client1]`	Step 4
Registered the key client to the tenant	`c1Client1 to c1Tenant1`	`[c2Client1 to c1Tenant1]`	Step 5

Table 53. Setup of Cluster1 and Cluster2 (continued)

Item	Cluster1	Cluster2	Steps
Has access to master encryption keys	`c1Client1`	`[c2Client1]`	Step 6
Has access to encrypted file	Local access to `hw.enc` in `c1FileSystem1`	[Remote access to `hw.enc` in `c1FileSystem1`.]	Step 6

The encrypted file `hw.enc` is in `c1FileSystem1` on `Cluster1`. To configure `Cluster2` to have remote access to file `hw.enc`, follow these steps:

1. From a node in `Cluster2`, connect to the remote `Cluster1`:

   a. To set up access to the remote cluster and file system, follow the instructions in topic Chapter 24, "Accessing a remote GPFS file system," on page 339.

   b. Run the **mmremotefs add** command to make the remote file system `c1FileSystem1` known to the local cluster, `Cluster2`:

   Note: `c1Filesystem1_Remote` is the name by which the remote file system `c1FileSystem1` is known to `Cluster2`.

   ```
   # mmremotefs add c1FileSystem1_Remote -f c1FileSystem1 -C Cluster1.gpfs.net -T
   /c1FileSystem1_Remote -A no
   mmremotefs: Propagating the cluster configuration data to all affected nodes.
       This is an asynchronous process.
   Tue Mar 29 06:38:07 EDT 2016: mmcommon pushSdr_async: mmsdrfs propagation started.
   ```

   Note: After you have completed Step 1(b) and mounted the remote file system, if you try to access the contents of file `hw.enc` from `Cluster2`, the command fails because the local cluster does not have the master encryption key for the file:

   ```
   # cat /c1FileSystem1_Remote/hw.enc
   cat: hw.enc: Operation not permitted

   mmfs.log:
   Tue Mar 29 06:39:27.306 2016: [E]
   Key 'KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1:keyserver01_devG1'
   could not be fetched. The specified RKM ID does not exist;
   check the RKM.conf settings.
   ```

2. From a node in `Cluster2`, connect to the same SKLM key server, `keyserver01`, that `Cluster1` is connected to:

   a. Run the **mmkeyserv server add** to connect to `keyserver01`:

   ```
   # mmkeyserv server add keyserver01
   Enter password for the key server keyserver01:
   The security certificate(s) from keyserver01.gpfs.net must be accepted to continue.

   View the certificate(s) to determine whether you want to trust the certifying authority.
   Do you want to view or trust the certificate(s)? (view/yes/no) view

   Serial number:          01022a8adf20f3
   SHA-256 digest:         2ca4a48a3038f37d430162be8827d91eb584e98f5b3809047ef4a1c72e15fc4c
   Signature:              7f0312e7be18efd72c9d8f37dbb832724859ba4bb5827c230e2161473e0753b367ed49d
   993505bd23858541475de8e021e0930725abbd3d25b71edc8fc3de20b7c2db5cd4e865f41c7c410c1d710acf222e1c4
   5189108e40568ddcbeb21094264da60a1d96711015a7951eb2655363309d790ab44ee7b26adf8385e2c210b8268c5ae
   de5f82f268554a6fc22ece6efeee2a6264706e71416a0dbe8c39ceacd86054d7cc34dda4fffea4605c037d321290556
   10821af85dd9819a4d7e4baa70c51addcda720d33bc9f8bbde6d292c028b2f525a0275ebea968c26f8f0c4b604719ae
   3b04e71ed7a8188cd6adf68764374b29c91df3d101a941bf8b7189485ad72
   Signature algorithm:    SHA256WithRSASignature
   Key size:               2048
   Issuer:        C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net
   Subject:       C=US, O=IBM, OU=SKLMNode, SKLMCell, CN=c40bbc1xn3.gpfs.net

   Serial number:          01022a24475466
   ```

```
SHA-256 digest:         077c3b53c5046aa893b760c11cca3a993efbc729479771e03791f9ed4f716879
Signature:              227b5befe89f2e55ef628da6b50db1ab842095a54e1505655e3d95fee753a7f7554868a
a79b294c503dc34562cf69c2a20128796758838968565c0812c4aedbb0543d396646a269c02bf4c5ce5acba4409a10e
ffbd47ca38ce492698e2dcdc8390b9ae3f4a47c23ee3045ff0145218668f35a63edac68201789ed0db6e5c170f5c6db
49769f0b4c9a5f208746e4342294c447793ed087fa0ac762588faf420febeb3fca411e4e725bd46476e1f9f44759a69
6573af5dbbc9553218c7083c80440f2e542bf56cc5cc18156cce05efd6c2e5fea2b886c5c1e262c10af18b13ccf38c3
533ba025b97bbe62f271545b2ab5c1f50c1dca45ce504dfcfc257362e9b43
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:         C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net
Subject:        C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=c40bbc1xn3.gpfs.net

Do you trust the certificate(s) above? (yes/no) yes
```

b. Verify that the connection succeeded:

```
# mmkeyserv server show
keyserver01.gpfs.net
        Type:           ISKLM
        IPA:            192.168.40.59
        User ID:        SKLMAdmin
        REST port:      9080
        Label:          1_keyserver01
        NIST:           on
        FIPS1402:       off
        Backup Key Servers:
        Distribute:     yes
        Retrieval Timeout:  120
        Retrieval Retry:    3
        Retrieval Interval: 10000
```

3. From a node in Cluster2, add the same tenant, c1Tenant1, that Cluster1 added:

a. Add the tenant devG1:

```
# mmkeyserv tenant add devG1 --server keyserver01
Enter password for the key server keyserver01:
mmkeyserv: [I] Tenant devG1 belongs to GPFS family exists on the key server.
Processing continues ...
mmkeyserv: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

b. Verify that the tenant is added:

```
# mmkeyserv tenant show
devG1
        Key Server:         keyserver01.gpfs.net
        Registered Client:  (none)
```

4. From a node in Cluster2, create a key client:

a. Create the key client c2Client1:

```
 # mmkeyserv client create c2Client1 --server keyserver01
Enter password for the key server keyserver01:
Create a pass phrase for keystore:
Confirm your pass phrase:
mmkeyserv: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

b. Verify that the key client is created:

```
# mmkeyserv client show
c2Client1
        Label:          c2Client1
        Key Server:     keyserver01.gpfs.net
        Tenants:        (none
```

5. From a node in Cluster2, register the key client to the same tenant that Cluster1 is registered to. The RKM ID must be the same as the one that Cluster1 uses, to allow files created with that RKM ID on Cluster1 to be accessed from Cluster2. However, some of the information in the RKM stanza is different:

a. Register the client in Cluster2 to the same tenant c1Tenant1:

```
# mmkeyserv client register c2Client1 --tenant devG1 --rkm-id keyserver01_devG1
Enter password for the key server :
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.
```

b. Verify that the tenant shows that c2Client1 is registered:

```
# mmkeyserv tenant show
devG1
        Key Server:           keyserver01.gpfs.net
        Registered Client:    c2Client1
```

c. Verify that c2Client1 shows that it is registered to the c1Tenant:

```
# mmkeyserv client show
c2Client1
        Label:              c2Client1
        Key Server:         keyserver01.gpfs.net
        Tenants:            devG1
```

d. You can display the contents of the new RKM stanza:

```
 # mmkeyserv rkm show
keyserver01_devG1 {
  type = ISKLM
  kmipServerUri = tls://192.168.40.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c2Client1.1.p12
  passphrase = c2Client1
  clientCertLabel = c2Client1
  tenantName = devG1
}
```

e. You can also view the RKM stanza by displaying the contents of the RKM.conf file on the command-line console:

```
# cat /var/mmfs/ssl/keyServ/RKM.conf
keyserver01_devG1 {
type = ISKLM
kmipServerUri = tls://192.168.40.59:5696
keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c2Client1.1.p12
passphrase = pwc2Client1
clientCertLabel = c2Client1
tenantName = devG1
}
```

6. You can now access the encrypted file hw.enc remotely from Cluster2:

a. Verify that you can access the contents of the file hw.enc:

```
# cat /c1FileSystem1_Remote/hw.enc
Hello World!
```

b. Display the encryption attributes of the file:

```
# mmlsattr -n gpfs.Encryption /c1FileSystem1_Remote/hw.enc
file name: /c1FileSystem1_Remote/hw.enc
gpfs.Encryption: "EAGC????t!?v??????????? ???????=T??????????????? ???O?3???)??r??nV?K?OA?;????
??x,?:w?d???5)?KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1?keyserver01_devG1?"
EncPar 'AES:256:XTS:FEK:HMACSHA512'
type: wrapped FEK
WrpPar 'AES:KWRAP'
CmbPar 'XORHMACSHA512'
KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1:keyserver01_devG1
```

You can now access encrypted files on c1FileSystem1_Remote from Cluster2.

Simplified setup: Doing other tasks

Learn how to do other tasks after you complete the simplified setup.

For the first three tasks in this topic, you need the password for your SKLM key server.

"Creating encryption keys"

"Adding a tenant"

"Managing another key server" on page 592

"Adding backup key servers" on page 596

Creating encryption keys

This task shows how to create encryption keys in a tenant:

1. The following command creates five encryption keys in tenant devG1 on key server keyserver01 and displays the UUIDs of the keys on the console:

```
# mmkeyserv key create --server keyserver01.gpfs.net --tenant devG1 --count 5
Enter password for the key server keyserver01.gpfs.net:
KEY-492911c8-e3d4-4670-9868-617243d4ca57
KEY-5f24d71f-daf3-4df8-90e4-5f6475370f70
KEY-a487b01d-f092-4895-b537-139edeb57239
KEY-b449b3a2-73c5-499f-b575-fc7ba95541a8
KEY-fd3dbee9-0e6c-4662-9410-bfe3b73272b9
```

2. The following command shows the UUIDs of the encryption keys on tenant devG1 in keyserver01:

```
# mmkeyserv key show --server keyserver01.gpfs.net --tenant devG1
Enter password for the key server keyserver01.gpfs.net:
KEY-492911c8-e3d4-4670-9868-617243d4ca57
KEY-5f24d71f-daf3-4df8-90e4-5f6475370f70
KEY-a487b01d-f092-4895-b537-139edeb57239
KEY-b449b3a2-73c5-499f-b575-fc7ba95541a8
KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1
KEY-fd3dbee9-0e6c-4662-9410-bfe3b73272b9
```

The command displays the UUIDs of the previously existing key and the five new keys.

Adding a tenant

A tenant is a container that resides on a key server and contains encryption keys. Before a key client can request master encryption keys from a key server, you must add a tenant to the key server, create a key client, and register the key client with the tenant. For more information, see "Simplified setup: Using SKLM with a self-signed certificate" on page 568.

In some situations, you might need to access more than one tenant on the same key server. For example, if you have several key clients that you want to use with the same key server, each key client must register with a different tenant. For more information, see "Simplified setup: Valid and invalid configurations" on page 584.

This task shows how to add a tenant, register an existing key client with the tenant, and create encryption keys in the tenant.

1. Add the tenant:

   a. Add a tenant devG2 on keyserver01:

   ```
   # mmkeyserv tenant add devG2 --server keyserver01
   Enter password for the key server keyserver01:
   ```

   b. Verify that the tenant is added. The following command displays all the existing tenants:

   ```
   # mmkeyserv tenant show
   devG1
           Key Server:          keyserver01.gpfs.net
           Registered Client:   c1Client1
   ```

```
devG2
        Key Server:        keyserver01.gpfs.net
        Registered Client:  (none)
```

The tenants are `devG1` and `devG2`.

2. Register the existing key client with the tenant:

   a. Register client `c1Client1` with tenant `devG2`:

   ```
   # mmkeyserv client register c1Client1 --tenant devG2 --rkm-id keyserver01_devG2
   Enter password for the key server :
   mmkeyserv: [I] Client currently does not have access to the key.
   Continue the registration process...
   mmkeyserv: Successfully accepted client certificate
   ```

   b. Verify that the key client is registered to the tenant:

   ```
   # mmkeyserv client show
   c1Client1
           Label:            c1Client1
           Key Server:       keyserver01.gpfs.net
           Tenants:          devG1,devG2
   ```

   The command output shows that `c1Client1` is registered to both `devG1` and the new `devG2`.

   c. Verify the configuration of the RKM stanza. The following command displays all the RKM stanzas:

   ```
   # mmkeyserv rkm show
   keyserver01_devG1 {
     type = ISKLM
     kmipServerUri = tls://192.168.40.59:5696
     keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
     passphrase = pw_c1Client1
     clientCertLabel = label_c1Client1
     tenantName = devG1
   }
   keyserver01_devG2 {
     type = ISKLM
     kmipServerUri = tls://192.168.40.59:5696
     keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
     passphrase = pw_c1Client1
     clientCertLabel = label_c1Client1
     tenantName = devG2
   }
   ```

   The command shows the following relationships:

   * Client `c1Client1` is registered with tenant `devG1` on `keyserver01`.
   * Client `c1Client1` is also registered with tenant `devG2` on `keyserver01`.

3. Create keys in the tenant. The following command creates three keys in tenant `devG2`:

   ```
   # mmkeyserv key create --server keyserver01 --tenant devG2 --count 3
   Enter password for the key server keyserver01:
   KEY-43cf5e69-1640-4056-b114-bdbcf2914189
   KEY-4c7540cd-0346-4733-90eb-8df4c0f16008
   KEY-c86a523b-e04f-4536-86a6-c6f83f845265
   ```

Managing another key server

This task shows how to add a key server, add a tenant, create a new key client, and register the key client with the tenant. The steps are the same as the ones that you follow in the simplified setup:

Table 54. Managing another key server

Item	Step
Install and configure SKLM.	Step 1
Add a key server	Step 2

Table 54. Managing another key server (continued)

Item	Step
Add a tenant to the key server	Step 3
Create a key client	Step 4
Register the key client with the tenant	Step 5

1. Install and configure IBM Security Key Lifecycle Manager (SKLM). For more information, see the topic "Simplified setup: Using SKLM with a self-signed certificate" on page 568.

2. Add the key server, `keyserver11`. If backup key servers are available, you can add them now. You can have up to five backup key servers.

   a. Add `keyserver11` and backup key servers `keyserver12` and `keyserver13`. Enter the requested information when prompted:

```
# mmkeyserv server add keyserver11 --backup keyserver12,keyserver13
Enter password for the key server keyserver11:
The security certificate(s) from keyserver11.gpfs.net must be accepted to continue.
View the certificate(s) to determine whether you want to trust the certifying authority.
Do you want to view or trust the certificate(s)? (view/yes/no) view

Serial number:          0361e7075056
SHA-256 digest:         2a7ab79d52cca7d2cae6e88077ee48b405a9e87d03d47023fdf1d4e185f18f75
Signature:              55a4350778446ac1f74fe25016bc9efd86893b8c5e9a4c3ebc4662d7cafce8697bfbf98
f8ce62ab976fb10270a006074bd36a3c0321bb99417dcd6d9d18c06ca380f1a89aacf3d0b5d84a7fdde5d4c1b9377a0
e725d65dee819f489a9c51c2017ac6633304a3973c7e13ddc611aae6d2ba35c8571b6ca1388dbb1b91a51b00f09fe37
2846dbe0139e4f942ed317809c0b7d0cd651a3273b4df041719f99847923e5ec58517fd778d46ea44647149c5d52287
ee9705aa292c1d2942b27dd7f07d6bae2b1f29a4a818655c582ef0ce9102e70a7df68ee0c0732a66b2960959f38f964
0c599a3203ff6fcafc13f40e9922fa439d016937a00d0f5a7f571d174f277
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=vmip131.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, CN=vmip131.gpfs.net

Serial number:          03615d201517
SHA-256 digest:         4acb77202f885f4c6b4c858f701394f18150fd683a0d155885399bbb5b8cc0b1
Signature:              15e2011efd402b4834c677c9bcdca9914f457a9573bf1568c4d309cd1a9b873b857566c
f9653a736e34b63f8e600e1bee2450c838bbf49c6291548f0bb4ee82d8243ba60dcfbcc42f25f965fa36483441dfe7e
b2089361dbee77e333d2711ee8364f9d5005cf382a42fa90dec8f0e279b5cecb6d5ef3da2d75cdc1e70d7f4545afc13
547135c4978b717c6572b3d8c569cd44f15c0b084fe92a9e2878bcf34518882c1461e832e014d56d981ad40ef2c6760
71f49571a91e036c84ab58b3d22d0d971990624751ea6d74a420cfbf2e00d718e263184c97091404d295adb56467237
09decacebd7dbfa1927a8143bdf6d6640b72ec7c588b00cf0521c67f6efe9
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=vmip131.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate, CN=vmip131.gpfs.net

Do you trust the certificate(s) above? (yes/no) yes
```

   b. Verify that the key server is added. The following command displays information about all the existing key servers:

```
# mmkeyserv server show
keyserver01.gpfs.net
        Type:              ISKLM
        IPA:               192.168.40.59
        User ID:           SKLMAdmin
        REST port:         9080
        Label:             1_keyserver01
        NIST:              on
        FIPS1402:          off
        Backup Key Servers:
        Distribute:        yes
        Retrieval Timeout: 120
        Retrieval Retry:   3
        Retrieval Interval: 10000
```

```
keyserver11.gpfs.net
        Type:                 ISKLM
        IPA:                  192.168.9.131
        User ID:              SKLMAdmin
        REST port:            9080
        Label:                2_keyserver11
        NIST:                 on
        FIPS1402:             off
        Backup Key Servers:   keyserver12.gpfs.net,keyserver13.gpfs.net
        Distribute:           yes
        Retrieval Timeout:    120
        Retrieval Retry:      3
        Retrieval Interval:   10000
```

The command shows two key servers, keyserver01 and the keyserver11.

3. Add a tenant to the key server. The name of the tenant must be unique within the same key server, but it can be the same as the name of a tenant in another key server:

   a. Add the tenant devG1 to keyserver11:

   ```
   mmkeyserv tenant add devG1 --server keyserver11
   Enter password for the key server keyserver11:
   ```

   b. Verify that the tenant is added:

   ```
   mmkeyserv tenant show
   devG1
           Key Server:           keyserver01.gpfs.net
           Registered Client:    c1Client1

   devG2
           Key Server:           keyserver01.gpfs.net
           Registered Client:    c1Client1

   devG1
           Key Server:           keyserver11.gpfs.net
           Registered Client:    (none)
   ```

   The command shows the following tenants:

   • Tenant devG1 on keyserver01.
   • Tenant devG2 on keyserver01.
   • Tenant devG1 on keyserver11.

4. Create a key client:

   Note: A key client name must be 1-16 characters in length and must be unique within an IBM Spectrum Scale cluster.

   a. Create c1Client11 on keyserver11.

   ```
   # mmkeyserv client create c1Client11 --server keyserver11
   Enter password for the key server keyserver11:
   Create a pass phrase for keystore:
   Confirm your pass phrase:
   ```

   b. Verify that the client is created. The command shows all the existing key clients:

   ```
   # mmkeyserv client show
   c1Client1
           Label:                c1Client1
           Key Server:           keyserver01.gpfs.net
           Tenants:              devG1,devG2

   c1Client11
           Label:                c1Client11
           Key Server:           keyserver11.gpfs.net
           Tenants:              (none)
   ```

The key clients are c1Client1 and c1Client11.

c. You can also display all the clients of keyserver11:

```
# mmkeyserv client show --server keyserver11
c1Client11
        Label:            c1Client11
        Key Server:       keyserver11.gpfs.net
        Tenants:          (none)
```

5. Register the key client with the tenant:

a. Verify that tenant devG1 on keyserver11 has no registered clients:

```
# mmkeyserv tenant show --server keyserver11
devG1
        Key Server:       keyserver11.gpfs.net
        Registered Client:   (none)
```

b. Register the key client c1Client11 with the devG1 on keyserver11:

```
# mmkeyserv client register c1Client11 --tenant devG1 --rkm-id keyserver11_devG1
Enter password for the key server of client c1Client11:
mmkeyserv: [I] Client currently does not have access to the key.
Continue the registration process ...
mmkeyserv: Successfully accepted client certificate
```

c. Verify that the tenant shows that the client c1Client11 is registered with it:

```
 # mmkeyserv tenant show --server keyserver11
devG1
        Key Server:       keyserver11.gpfs.net
        Registered Client:   c1Client11
```

d. You can also verify that the client shows that it is registered with tenant devG1:

```
# mmkeyserv client show --server keyserver11
c1Client11
        Label:            c1Client11
        Key Server:       keyserver11.gpfs.net
        Tenants:          devG1
```

e. Display the RKM stanzas for the cluster. They show the following relationships:

   • With keyserver01, c1Client1 is registered with devG1 and devG2.

   • With keyserver11, c1Client11 is registered with devG1.

```
# mmkeyserv rkm show
keyserver01_devG1 {
  type = ISKLM
  kmipServerUri = tls://192.168.40.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
  passphrase = pw4c1Client1
  clientCertLabel = c1Client1
  tenantName = devG1
}
keyserver01_devG2 {
  type = ISKLM
  kmipServerUri = tls://192.168.40.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
  passphrase = pw4c1Client1
  clientCertLabel = c1Client1
  tenantName = devG2
}
keyserver11_devG1 {
  type = ISKLM
  kmipServerUri = tls://keyserver12.gpfs.net:5696
  kmipServerUri2 = tls://keyserver13.gpfs.net:5696
  kmipServerUri3 = tls://192.168.9.131:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.2_keyserver11.c1Client11.1.p12
  passphrase = pw4c1Client11
  clientCertLabel = c1Client11
  tenantName = devG1
}
```

f. Create encryption keys. The following command creates two keys in tenant devG1 on keyserver11.

```
# mmkeyserv key create --server keyserver11 --tenant devG1 --count 2
Enter password for the key server keyserver11:
KEY-86f601ba-0643-4f94-92b2-12c8765512cc
KEY-cdcf058f-ae30-41e8-b6f7-754e23322428
```

Adding backup key servers

If multiple key servers exist, you can add them to an RKM stanza to provide backup capability in case the main key server becomes unavailable. You can add up to five backup key servers.

Important: IBM Spectrum Scale does not manage backup key servers. You must configure them and maintain them.

This task shows how to add backup key servers to the RKM stanza of one of your key clients. You can add backup key servers when you create a key server, as shown in Step 2 of the previous subtopic. Or you can add them later, as in this subtopic.

In this task the primary key server is keyserver11. The backup key servers for the RKM stanza are keyserver12 and keyserver13. You want to add three more backup key servers to the list: keyserver14, keyserver15, and keyserver16.

Follow these steps:

1. Add the three backup key servers. You must specify the entire list of key servers, including ones that are already in the list. The following command is on one line. In the list of servers, do not put spaces on either side of the commas (,):

   ```
   # mmkeyserv rkm change keyserver11_devG1 --backup
   keyserver12,keyserver13,keyserver14,keyserver15,keyserver16
   ```

   Attention:
   • You can change the order in which the client tries backup key servers, by running the same command with the key servers in a different order.
   • You can delete backup key servers by specifying a list that contains the backup key servers that you want to keep and omits the ones that you want to delete.

2. To verify, issue the **mmkeyserv rkm show** command to display the RKM stanzas:

   ```
   # mmkeyserv rkm show
   keyserver01_devG1 {
     type = ISKLM
     kmipServerUri = tls://192.168.40.59:5696
     keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
     passphrase = pw4c1Client1
     clientCertLabel = c1Client1
     tenantName = devG1
   }

   keyserver01_devG2 {
     type = ISKLM
     kmipServerUri = tls://192.168.40.59:5696
     keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c1Client1.1.p12
     passphrase = pw4c1Client1
     clientCertLabel = c1Client1
     tenantName = devG2
   }
   keyserver11_devG1 {
     type = ISKLM
     kmipServerUri = tls://keyserver11.gpfs.net:5696
     kmipServerUri12 = tls://keyserver12.gpfs.net:5696
     kmipServerUri13 = tls://keyserver13.gpfs.net:5696
     kmipServerUri14 = tls://keyserver14.gpfs.net:5696
     kmipServerUri15 = tls://keyserver15.gpfs.net:5696
     kmipServerUri16 = tls://keyserver16.gpfs.net:5696
     keyStore = /var/mmfs/ssl/keyServ/serverKmip.2_keyserver11.c1Client11.1.p12
   ```

```
    passphrase = pw4c1Client11
    clientCertLabel = c1Client11
    tenantName = devG1
}
```

The command outputs shows the following relationships:
- The configuration of `c1Client1`, `devG1`, and `keyserver01` has zero backup servers.
- The configuration of `c1Client1`, `devG2`, and `keyserver01` has zero backup servers.
- The configuration of `c1Client11`, `devG1`, and `keyserver11` has five backup servers.

Regular setup: Using SKLM with a self-signed certificate

The regular setup with IBM Security Key Lifecycle Manager (SKLM) requires IBM Spectrum Scale Advanced Edition or IBM Spectrum Scale Data Management Edition V4.1 or later and SKLM V2.5.0.1 or later (including V2.6).

This topic describes the regular setup for setting up encryption with IBM Security Key Lifecycle Manager (SKLM) as the key management server and using self-signed certificates on the KMIP port of the SKLM server. If your deployment scenario requires the use of a chain of server certificates from a Certificate Authority, see the topic "Regular setup: Using SKLM with a certificate chain" on page 604.

Note: If you are using SKLM v2.7 or later, see the topic "Configuring encryption with SKLM v2.7" on page 614.

Requirements:

The following requirements must be met on every IBM Spectrum Scale node that you configure for encryption:
- The node must have direct network access to the system where the key server is installed.
- The security-sensitive files that are created during the configuration process must have the following characteristics:
  - They must be regular files that are owned by the root user.
  - They must be in the root group.
  - They must be readable and writable only by the user (mode '0600'). The following examples apply to the regular setup and the Vormetric DSM setup:
    ```
    -rw-------. 1 root root 2446 Mar 20 12:15 /var/mmfs/etc/RKM.conf
    drw-------. 2 root root 4096 Mar 20 13:47 /var/mmfs/etc/RKMcerts
    -rw-------. 1 root root 3988 Mar 20 13:47 /var/mmfs/etc/RKMcerts/keystore_name.p12
    ```

This security-sensitive files includes the following files:
  - The `RKM.conf` file. For more information about this file, see "The RKM.conf file and the RKM stanza" on page 565.
  - The files in the client keystore directory, which include the keystore file, the public and private key files for the client, and possibly other files. For more information about these files, see "The client keystore directory and its files" on page 567.

  CAUTION:
  - **Take appropriate precautions to ensure that the security-sensitive files are not lost or corrupted. IBM Spectrum Scale does not manage or replicate the files.**
  - **Ensure that the passphrase for the client certificate file is not leaked through other means, such as the shell history.**
- Client keystore files must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version of NFS or

store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

See the following subtopics for instructions:

"Part 1: Installing Security Key Lifecycle Manager"

"Part 2: Creating and exporting a server certificate" on page 599

"Part 3: Configuring the remote key management (RKM) back end" on page 600

"Part 4: Configuring more RKM back ends" on page 604

Part 1: Installing Security Key Lifecycle Manager

Follow the instructions in this subtopic to install and configure the IBM Security Key Lifecycle Manager (SKLM).

1. Install IBM Security Key Lifecycle Manager. For the required version, see "Preparation for encryption" on page 563. For the installation, choose a system that the IBM Spectrum Scale node that you want to configure has direct network access to. For information about installing SKLM, see the *Installing and configuring* chapter of the SKLM documentation.

2. From the main page of the SKLM web GUI, click **Configuration** > **Key Serving Parameters** and select the check box for **Keep pending client device communication certificates**.

3. Configure SKLM to have the same FIPS 140-2 (FIPS) setting as the IBM Spectrum Scale cluster. Follow these steps:

   a. Determine the FIPS setting of the cluster by entering the following command on the command line:

      ```
      mmlsconfig FIPS1402mode
      ```

      The command returns `yes` if the cluster complies with FIPS or `no` if not.

   b. On the SKLM server system, open the `SKLMConfig.properties` file.

      Note: The default location of the `SKLMConfig.properties` file depends on the operating system:

      • On AIX, Linux, and similar operating systems:

        ```
        /opt/IBM/WebSphere/AppServer/products/sklm/config/SKLMConfig.properties
        ```

      • On Microsoft Windows:

        ```
        Drive:\Program Files (x86)\IBM\WebSphere\AppServer\products\sklm\config\
        SKLMConfig.properties
        ```

   c. Add or remove the following line from the `SKLMConfig.properties` file. Add the line to configure SKLM to comply with FIPS, or remove it to have SKLM not comply with FIPS.

      ```
      fips=on
      ```

4. Configure the SKLM server to have the same NIST SP800-131a (NIST) setting as the IBM Spectrum Scale cluster. Follow these steps:

   a. Determine the NIST setting of the cluster by entering the following command on the command line:

      ```
      mmlsconfig nistCompliance
      ```

      The command returns `SP800-131A` if the cluster complies with NIST or `off` if not.

   b. On the SKLM server system, open the `SKLMConfig.properties` file. For the location of this file, see the note in Step 3.

   c. Add the following line to configure SKLM to comply with NIST or remove it to configure SKLM not to comply with NIST:

      ```
      TransportListener.ssl.protocols=TLSv1.2
      ```

   d. For all V2.5.0.x versions of SKLM, if you are configuring SKLM to comply with NIST, modify the following variable to include only cipher suites that are approved by NIST. The following statement is all on one line, with no space before or after the comma (,):

```
TransportListener.ssl.ciphersuites=TLS_RSA_WITH_AES_256_CBC_SHA256,
    TLS_RSA_WITH_AES_128_CBC_SHA256
```

5. If the cipher suites have been set at any time, SKLM 2.6.0.0 has a known issue that causes server certificates always to be signed with SHA1withRSA. To work around the problem, follow these steps:

   a. While the SKLM server is running, in the `SKLMConfig.properties` file, add or modify the `requireSHA2Signatures` property as follows:

      `requireSHA2Signatures=true`

   b. Do not restart the server.

   c. Generate a new server certificate and set it to be the one in use.

   d. If you restart the server, you must repeat this workaround before you can create a server certificate that is signed other than with SHA1withRSA.

Part 2: Creating and exporting a server certificate

Follow the instructions in this subtopic to create and export a server certificate in SKLM:

1. Create a self-signed server certificate:

   a. On the system where SKLM is running, open the graphical user interface.

   b. Click **Configuration** > **SSL/KMIP**.

   c. Click **Create self-signed certificate**.

   d. Enter the information for the certificate and click **OK**.

   e. Restart the server to verify that the server can operate with the new certificate.

2. Make a note of the label of the certificate that is in use:

   a. In the SKLM graphical user interface, click **Advanced Configuration** > **Server Certificates**.

   b. Select the certificate that is identified as being in use. Click **Modify** and make a note of the certificate label. You need it in Step 3.

3. Export the certificate through the command-line interface. Follow these steps:

   a. On the SKLM server system, open a command-line window.

   b. Change to the *WAS_HOME*/bin directory. The location of this directory depends on the operating system:

      • On AIX, Linux, and similar operating systems:

         `/opt/IBM/WebSphere/AppServer/bin`

      • On Microsoft Windows:

         `drive:\Program Files (x86)\IBM\WebSphere\AppServer\bin`

   c. Enter the following command to start the command-line interface to SKLM:

      • On AIX, Linux, and similar operating systems:

         `./wsadmin.sh -username SKLMAdmin -password mypwd -lang jython`

      • On Microsoft Windows:

         `wsadmin -username SKLMAdmin -password mypwd -lang jython`

   d. In the SKLM command line interface, enter the following command:

      `print AdminTask.tklmCertList('[-alias labelSSCert]')`

      where:

      labelSSCert
         Specifies the certificate label of the self-signed server certificate. You made a note of the label in Step 2.

      SKLM responds with output like the following example:

      ```
      CTGKM0001I Command succeeded.
      uuid = CERTIFICATE-7005029a-831d-405f-af30-4bf0177909de
      alias = server
      key store name = defaultKeyStore
      ```

```
key state = ACTIVE
issuer name = CN=server
subject name = CN=server
creation date = 13/03/2014 16:27:13 Eastern Daylight Time
expiration date = 09/03/2015 07:12:30 Eastern Daylight Time
serial number = 1394363550
```

    e. Make a note of the UUID of the certificate that is displayed on line 2 of the output. You need it in the next substep and in Part 3.

    f. To export the certificate, from the SKLM command line interface, enter the following command on one line:

```
print AdminTask.tklmCertExport('[-uuid certUUID -format base64 -fileName fileName]')
```

       where:

       certUUID
            Specifies the UUID that you made a note of in the previous substep.

       fileName
            Specifies the path and file name of the certificate file in which the server certificate is stored.

       SKLM exports the self-signed server certificate into the specified file.

    g. Close the SKLM command line interface.

    h. Copy the server certificate file to a temporary directory on the IBM Spectrum Scale node that you are configuring for encryption.

4. In SKLM, create a device group and keys for the IBM Spectrum Scale cluster:

    a. In the SKLM graphical user interface, click **Advanced Configuration** > **Device Group**.

    b. In the Device Group table, click **Create**.

    c. In the Create Device Group window, follow these steps:

      1) Select the **GPFS** device family.

      2) Enter an appropriate name, such as `GPFS_Tenant0001`. The name is case-sensitive.

      3) Make a note of the name. You need it in Part 3 when you create an RKM stanza.

      4) Complete any other fields and click **Create**.

    d. After SKLM creates the device group, it prompts you to add devices and keys. Do not add any devices or keys. Instead, click **Close**. Keys are created in the next step.

5. Create keys for the device group.

    a. In the SKLM graphical user interface, in the Key and Device Management table, select the device group that you created in Step 4. In these instructions the device group is named `GPFS_Tenant0001`.

    b. Click **Go to** > **Manage keys and services**.

    c. In the management page for `GPFS_Tenant0001`, click **Add** > **Key**.

    d. Enter the following information:

      • The number of keys to be created

      • The three-letter prefix for key names. The key names are internal SKLM names and are not used for GPFS encryption.

    e. Make a note of the UUID of the key, such as `KEY-326a1906-be46-4983-a63e-29f005fb3a15`. You need it in Part 3.

    f. In the drop-down list at the bottom of the page, select **Hold new certificate requests pending my approval**.

Part 3: Configuring the remote key management (RKM) back end

An RKM back end defines a connection between a local key client, a remote key tenant, and an RKM server. Each RKM back end is described in an RKM stanza in an `RKM.conf` file on each node that is configured for encryption.

This subtopic describes how to configure a single RKM back end and how to share the configuration among multiple nodes in a cluster. To configure multiple RKM back ends, see "Part 4: Configuring more RKM back ends" on page 604

You can do Step 1 and Step 2 on any node of the cluster. In later steps you will copy the configuration files from Step 1 and Step 2 to other nodes in the cluster.

1. Create and configure a client keystore. Follow these steps:
    a. Create the following subdirectory to contain the client keystore:

       /var/mmfs/etc/RKMcerts

    b. The following command creates the client keystore, stores a private key and a client certificate in it, and also stores the trusted SKLM server certificate into it. From the command line, enter the following command on one line:

       mmauth gencert --cname *clientName* **--cert** *serverCertFile* **--out** */var/mmfs/etc/RKMcerts/SKLM.p12*
       --label *clientCertLabel* **--pwd-file** *passwordFile*

       where the parameters are as follows:

       --cname *clientName*
              The name of the client that is used in the certificate.

       --cert *serverCertFile*
              The path and file name of the file that contains the SKLM server certificate. You extracted this certificate from SKLM and copied the certificate file to the node in Part 2, Step 3h.

       --out */var/mmfs/etc/RKMcerts/SKLM.p12*
              The path and file name of the client keystore.

       --label *clientCertLabel*
              The label of the client certificate in the keystore. The label can be 1-20 characters in length.

       --pwd-file *passwordFile*
              The path of a text file that contains the password for the client keystore. The password can be 1-20 characters in length.

    Important: Verify that the files in the client keystore directory meet the requirements for security-sensitive files that are listed in the Requirements section at the beginning of this topic.

2. Create an RKM.conf file and add a stanza to it that describes a connection between a local key client, an SKLM device group, and an SKLM key server. Each stanza defines an RKM back end.
    a. Create a text file with the following path and name:

       /var/mmfs/etc/RKM.conf

       Important: Verify that the files in the client keystore directory meet the requirements for security-sensitive files that are listed in the Requirements section at the beginning of this topic.

    b. Add a stanza with the following format:

       ```
       stanzaName {
          type = ISKLM
          kmipServerUri = tls://raclette.zurich.ibm.com:5696
          keyStore = /var/mmfs/etc/RKMcerts/SKLM.p12
          passphrase = a_password
          clientCertLabel = a_label
          tenantName = GPFS_Tenant0001
       }
       ```

       where the rows of the stanza have the following meaning:

       stanzaName
              A name (RKM ID) for the stanza. Make a note of the name: you need it in the next step.

It is a good practice to use a format like the following one to ensure that the RKM ID is unique:

keyServerName_tenantName

where *tenantName* is the name that you provide in the last line of stanza. For example, the RKM ID for the key server and key client in these instructions is: `raclette_GPFS_Tenant0001`.

type Always ISKLM.

kmipServerUri

The DNS name or IP address of the SKLM server and the KMIP SSL port. You can find this information on the main page of the SKLM graphic user interface. The default port is 5696.

You can have multiple instances of this line, where the first instance represents the primary key server and each additional instance represents a backup key server. You can have up to five backup key servers. The following example has the primary key server and five backup key servers:

```
stanzaName {
    type = ISKLM
    kmipServerUri = tls://raclette.zurich.ibm.com:5696
    kmipServerUri2 = tls://raclette.fondue2.ibm.com:5696
    kmipServerUri3 = tls://raclette.fondue3.ibm.com:5696
    kmipServerUri4 = tls://raclette.fondue4.ibm.com:5696
    kmipServerUri5 = tls://raclette.fondue5.ibm.com:5696
    kmipServerUri6 = tls://raclette.fondue6.ibm.com:5696
    keyStore = /var/mmfs/etc/RKMcerts/SKLM.p12
    passphrase = a_password
    clientCertLabel = a_label
    tenantName = GPFS_Tenant0001
}
```

If the GPFS daemon cannot get an encryption key from the primary key server, it tries the backup key servers in order.

For more information, see "RKM back ends" on page 565 in "Preparation for encryption" on page 563.

keyStore

The path and name of the client keystore. You specified this parameter in Step 1.

passphrase

The password of the client keystore and client certificate. You specified this parameter in Step 1.

clientCertLabel

The label of the client certificate in the client keystore. You specified this parameter in Step 1.

tenantName

The name of the SKLM device group. See "Part 1: Installing Security Key Lifecycle Manager" on page 598.

3. Copy the configuration files to the file system manager node:

Note: The `mmchpolicy` command in Step 5 will fail if you omit this step. The `mmchpolicy` command requires the configuration files to be on the file system manager node.

a. Copy the `RKM.conf` file from the `/var/mmfs/etc` directory to the same directory on the file system manager node.

b. Copy the keystore files that the RKM file references to the same directories on the file system manager node. The recommended location for the keystore files is `/var/mmfs/etc/RKMcerts/`.

Important: Verify that the files in the client keystore directory meet the requirements for security-sensitive files that are listed in the Requirements section at the beginning of this topic.

4. To configure other nodes in the cluster for encryption, copy the `RKM.conf` file and the keystore files to those nodes. Copy the files in the same way as you did in Step 3.

   Important: Verify that the files in the client keystore directory meet the requirements for security-sensitive files that are listed in the Requirements section at the beginning of this topic.

5. Install an encryption policy for the cluster:

   Note: You can do this step on any node to which you copied the configuration files.

   a. Create a policy that instructs GPFS to do the encryption tasks that you want. The following policy is an example policy. It instructs IBM Spectrum Scale to encrypt all files in the file system with a file encryption key (FEK) and to wrap the FEK with a master encryption key (MEK):

   ```
   RULE 'p1' SET POOL 'system' # one placement rule is required at all times
   RULE 'Encrypt all files in file system with rule E1'
   SET ENCRYPTION 'E1'
   WHERE NAME LIKE '%'
   RULE 'simpleEncRule' ENCRYPTION 'E1' IS
   ALGO 'DEFAULTNISTSP800131A'
   KEYS('KEY-326a1906-be46-4983-a63e-29f005fb3a15:SKLM_srv')
   ```

   In the last line, the character string within single quotation marks (') is the key name. A *key name* is a compound of two parts in the following format:

   ```
   KeyID:RkmID
   ```

   where:

   KeyID
   Specifies the UUID of the key that you created in the SKLM graphic user interface in Part 2.

   RkmID
   Specifies the name of the RKM backend stanza that you created in the `/var/mmfs/etc/RKM.conf` file.

   b. Install the policy rule with the **mmchpolicy** command.

   Note: If an encryption policy succeeds on one node but fails on another node in the same cluster, verify that the failing node has the correct client keystore and stanza.

   CAUTION: Installing a new policy with the **mmchpolicy** command removes all the statements in the previous policy. To add statements to an existing policy without deleting the previous contents, collect all policy statements for the file system into one file. Add the new statements to the file and install the contents of the file with the **mmchpolicy** command.

6. Import the client certificate into the SKLM server:

   a. On the IBM Spectrum Scale node that you are configuring for encryption, send a KMIP request to SKLM. To send a KMIP request, try to create an encrypted file on the node. The attempt fails, but it causes SKLM to put the client certificate in a list of pending certificates in the SKLM key server. The attempt fails because SKLM does not yet trust the client certificate. See the following example:

   ```
   # touch /gpfs0/test
   touch: cannot touch `/gpfs0/test': Permission denied
   # tail -n 2 /var/adm/ras/mmfs.log.latest
   Thu Mar 20 14:00:55.029 2014: [E] Unable to open encrypted file: inode 46088,
   Fileset fs1, File System gpfs0.
   Thu Mar 20 14:00:55.030 2014: [E] Error: key
   'KEY-326a1906-be46-4983-a63e-29f005fb3a15:SKLM_srv' could not be fetched (RKM
   reported error -1004).
   ```

b. In the graphical user interface of SKLM, on the main page, click **Pending client device communication certificates.**

c. Find the client certificate in the list and click **View**.

d. Carefully check that the certificate that you are importing matches the one created in the previous step, then click **Accept and Trust**.

e. On the resulting screen, provide a name for the certificate and click **Accept and Trust** again.

f. On the node that you are configuring for encryption, try to create an encrypted file as you did in Step (a). This time the command succeeds. Enter an **mmlsattr** command to list the encryption attributes of the new file:

```
# touch /gpfs0/test
# mmlsattr -n gpfs.Encryption /gpfs0/test
file name: /gpfs0/test
gpfs.Encryption: "EAGC????f?????????????? ??????w?^??>???????????? ?L4??
_-???V}f???X????,?G?<sH??0?)??M?????)?KEY-326a1906-be46-4983-a63e-29f005fb3a15?
sklmsrv?)?KEY-6aaa3451-6a0c-4f2e-9f30-d443ff2ac7db?RKMKMIP3?"
EncPar 'AES:256:XTS:FEK:HMACSHA512'
type: wrapped FEK WrpPar 'AES:KWRAP' CmbPar 'XORHMACSHA512'
KEY-326a1906-be46-4983-a63e-29f005fb3a15:sklmsrv
```

From now on, the encryption policy rule causes each newly created file to be encrypted with a file encryption key (FEK) that is wrapped in a master encryption key (MEK). You created the key in a device group in the SKLM server and included its UUID as part of a key name in the security rule.l1

Part 4: Configuring more RKM back ends

To configure more RKM back ends, follow the steps in Part 3. You might want to:

• Add a primary or backup key server.

• Add a key client by creating and configuring a client keystore and importing the client certificate into the SKLM server.

• Define a back end by adding a stanza to the RLM.conf file. You can share client keystores, tenants, or key servers between stanzas.

Note the following design points:

– On a single node:

 - The RKM.conf file can contain multiple stanzas. Each stanza represents a connection between a key client and an SKLM device group.

 - You can create multiple keystores.

– Across different nodes:

 - The contents of RKM.conf files can be different.

 - The contents of keystores can be different.

 - If an encryption policy succeeds on one node and fails on another in the same cluster, verify that the failing node has the correct client keystore and stanza.

• Add encryption policies. Before you run the **mmchpolicy** command, ensure that the following conditions have been met:

– The keystore files and the RKM.conf files have been copied to the proper nodes.

– The files in the client keystore directory meet the requirements for security-sensitive files that are listed in the Requirements section at the beginning of this topic.

For more information, see "RKM back ends" on page 565 in "Preparation for encryption" on page 563.

Regular setup: Using SKLM with a certificate chain

Learn how to configure SKLM in the regular setup when you use a certificate chain from a certificate authority rather than a self-signed server certificate.

This topic describes the regular method for setting up encryption with IBM Security Key Lifecycle Manager (SKLM) as the key management server and with a certificate signed by a certificate authority (CA) on the KMIP port of the SKLM server. If your deployment scenario uses a self-signed server certificate, see one of the following topics:

"Simplified setup: Using SKLM with a self-signed certificate" on page 568

"Regular setup: Using SKLM with a self-signed certificate" on page 597

The regular setup with IBM Security Key Lifecycle Manager (SKLM) requires IBM Spectrum Scale Advanced Edition or IBM Spectrum Scale Data Management Edition V4.1 or later and SKLM V2.5.0.1 or later (including V2.6).

Note: If you are using SKLM v2.7 or later, see the topic "Configuring encryption with SKLM v2.7" on page 614.

Be aware of the following considerations:

The IBM Spectrum Scale node that you are configuring for encryption must have direct network access to the system where the key server is installed.

SECURITY NOTE: The contents of the following files are security-sensitive:
- The `RKM.conf` file
  - For the simplified setup, the file is at `/var/mmfs/ssl/keyServ/RKM.conf`
  - For the regular setup and the Vormetric DSM setup, the file is at `/var/mmfs/etc/RKM.conf`
- The directory for the client keystore
  - For the simplified setup, the directory is at `/var/mmfs/ssl/keyServ`
  - For the regular setup and the Vormetric DSM setup, the directory is at `/var/mmfs/etc/RKMcerts`
  - The temporary file that contains the client keystore and the client private key password.

IBM Spectrum Scale reads the contents of security-sensitive files only if the following conditions are met:
- They are regular files that are owned by the root user.
- They are in the root group.
- They are readable and writable only by the user.

See the permission bits in the following example:

```
-rw-------. 1 root root 2446 Mar 20 12:15 /var/mmfs/etc/RKM.conf
drw-------. 2 root root 4096 Mar 20 13:47 /var/mmfs/etc/RKMcerts
-rw-------. 1 root root 3988 Mar 20 13:47 /var/mmfs/etc/RKMcerts/keystore_name.p12
```

CAUTION:
It is a good practice to take the following precautions:
- **Ensure that the passphrase for the client certificate file is not leaked through other means, such as the shell history.**
- **Take appropriate precautions to ensure that the security-sensitive files are not lost or corrupted. IBM Spectrum Scale does not manage or replicate the files.**

Important: The client keystore must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version of NFS or store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

See the following subtopics for instructions:

"Part 1: Installing Security Key Lifecycle Manager" on page 606

Part 1: Installing Security Key Lifecycle Manager

Follow the instructions in this subtopic to install and configure the IBM Security Key Lifecycle Manager (SKLM).

1. Install IBM Security Key Lifecycle Manager. For the required version, see "Preparation for encryption" on page 563. For the installation, choose a system that the IBM Spectrum Scale node that you want to configure has direct network access to. For information about installing SKLM, see the *Installing and configuring* chapter of the SKLM documentation.

2. From the main page of the SKLM web GUI, click **Configuration** > **Key Serving Parameters** and select the check box for **Keep pending client device communication certificates**.

3. Configure SKLM to have the same FIPS 140-2 (FIPS) setting as the IBM Spectrum Scale cluster. Follow these steps:

   a. Determine the FIPS setting of the cluster by entering the following command on the command line:

      ```
      mmlsconfig FIPS1402mode
      ```

      The command returns `yes` if the cluster complies with FIPS or `no` if not.

   b. On the SKLM server system, open the `SKLMConfig.properties` file.

      Note: The default location of the `SKLMConfig.properties` file depends on the operating system:

      • On AIX, Linux, and similar operating systems the directory is at the following location:

         ```
         /opt/IBM/WebSphere/AppServer/products/sklm/config/SKLMConfig.properties
         ```

      • On Microsoft Windows the directory is at the following location:

         ```
         Drive:\Program Files (x86)\IBM\WebSphere\AppServer\products\sklm\config\
         SKLMConfig.properties
         ```

   c. Add or remove the following line from the `SKLMConfig.properties` file. Add the line to configure SKLM to comply with FIPS, or remove it to have SKLM not comply with FIPS.

      ```
      fips=on
      ```

4. Configure the SKLM server to have the same NIST SP800-131a (NIST) setting as the IBM Spectrum Scale cluster. Follow these steps:

   a. Determine the NIST setting of the cluster by entering the following command on the command line:

      ```
      mmlsconfig nistCompliance
      ```

      The command returns `SP800-131A` if the cluster complies with NIST or `off` if not.

   b. On the SKLM server system, open the `SKLMConfig.properties` file. For the location of this file, see the note in Step 3.

   c. Add the following line to configure SKLM to comply with NIST or remove it to configure SKLM not to comply with NIST:

      ```
      TransportListener.ssl.protocols=TLSv1.2
      ```

   d. For all V2.5.0.x versions of SKLM, if you are configuring SKLM to comply with NIST, modify the following variable to include only cipher suites that are approved by NIST. The following statement is all on one line, with no space before or after the comma:

      ```
      TransportListener.ssl.ciphersuites=TLS_RSA_WITH_AES_256_CBC_SHA256,
         TLS_RSA_WITH_AES_128_CBC_SHA256
      ```

5. Configure IBM WebSphere Application Server so that it has the same NIST setting as the IBM Spectrum Scale cluster. See the topic Transitioning WebSphere Application Server to the SP800-131 security standard in the volume *WebSphere Application Server Network Deployment* in the WebSphere Application Server online documentation.
   - WebSphere Application Server can be configured to run SP800-131 in a `transition` mode or a `strict` mode. The `strict` mode is recommended.
   - When NIST is enabled, make sure that WebSphere Application Server certificate size is at least 2048 bytes and is signed with SHA256withRSA as described in the preceding link.
6. If the cipher suites were set at any time, SKLM 2.6.0.0 has a known issue that causes server certificates always to be signed with SHA1withRSA. To work around the problem, follow these steps:
   a. While the SKLM server is running, in the `SKLMConfig.properties` file, modify the `requireSHA2Signatures` property as follows:

      `requireSHA2Signatures=true`

   b. Do not restart the server.
   c. Generate a new server certificate signing request (CSR) to a third-party certificate authority (CA) and send it to the CA.
   d. When you receive the certificate from the third-party CA, import it into SKLM and set it to be the certificate in use. For more information, see the next subtopic.
   e. If you restart the server, you must repeat this workaround before you can create a server certificate that is signed other than with SHA1withRSA.

Part 2: Configuring SKLM

To configure SKLM, you must create a certificate signing request (CSR), send it to the certificate authority (CA), obtain the certificate chain from the CA, and import the root certificate into the SKLM server. You must also create a device group for the cluster and create keys for the device group.

Note: For more information about the steps in this subtopic, see the steps that are described in the SKLM documentation, in the topic "Scenario: Request for a third-party certificate" at http://www.ibm.com/support/knowledgecenter/en/SSWPVP_2.7.0/com.ibm.sklm.doc/scenarios/cpt/cpt_ic_scenar_ca_certusage.html.

1. Create a certificate signing request (CSR) with the SKLM command-line interface:
   a. On the SKLM server system, open a command-line window.
   b.
   c. Change to the *WAS_HOME*/bindirectory. The location of this directory depends on the operating system:
      - On AIX, Linux, and similar operating systems, the directory is at the following location:

        `/opt/IBM/WebSphere/AppServer/bin`

      - On Microsoft Windows, the directory is at the following location:

        `drive:\Program Files (x86)\IBM\WebSphere\AppServer\bin`

   d. Start the command-line interface to SKLM:
      - On AIX, Linux, and similar operating systems, enter the following command:

        `./wsadmin.sh -username SKLMAdmin -password mypwd -lang jython`

      - On Microsoft Windows, enter the following command:

        `wsadmin -username SKLMAdmin -password mypwd -lang jython`

   e. In the SKLM command-line interface, enter the following command on one line:

      `print AdminTask.tklmCertGenRequest('[-alias labelCsr -cn server -validity daysValid -keyStoreName defaultKeyStore -fileName fileName -usage SSLSERVER]')`

      where:

-alias labelCsr
> Specifies the certificate label of the CSR.

-cn server
> Specifies the common name of the server in the certificate.

-validity daysValid
> Specifies the validity period of the certificate in days.

-keyStoreName defaultKeyStore
> Specifies the keystore name within SKLM where the CSR is stored. Typically, you should specify `defaultKeyStore` as the name here.

-fileName fileName
> Specifies the fully qualified path of the directory where the CSR is stored on the SKLM server system, for example `/root/sklmServer.csr`.

-usage SSLSERVER
> Specifies how the generated certificate is used in SKLM.

SKLM responds with output like the following example:
```
CTGKM0001I Command succeeded
fileName
```

2. Send the CSR file from Step 1 to the certificate authority.
3. When you receive the generated certificate file, or *endpoint certificate* file, from the certificate authority, copy it to a directory on the node that you are configuring for encryption. For example, you might copy it to the directory and file `/opt/IBM/WebSphere/AppServer/products/sklm/data/sklmServer.cert`.

   Important: You must also obtain and copy the intermediate certificate files of the certificate chain of authority into the same temporary directory. The intermediate certificates might be included with the generated certificate file, or you might have to obtain the intermediate certificates separately. Whatever the method, you must have a separate certificate file for the root certificate and for each intermediate certificate in the chain of authority. You need these certificate files in Part 3.

4. Import the root certificate into the SKLM server with the SKLM graphical user interface:
   a. On the **Welcome** page, in the **Action Items** section, in the **Key Groups and Certificates** area, click **You have pending certificates**.
   b. In the **Pending Certificates** table, click the certificate that you want to import and click**Import**.
   c. In the **File name and location** field, type the path and file name of the certificate file and click **Import**.

5. In SKLM, create a device group for the IBM Spectrum Scale cluster:
   a. In the SKLM graphical user interface, click **Advanced Configuration** > **Device Group**.
   b. In the **Device Group** table, click **Create**.
   c. In the Create Device Group window, follow these steps:
      1) Select the **GPFS** device family.
      2) Enter an appropriate name, such as `GPFS_Tenant0001`. The name is case-sensitive.
      3) Make a note of the name. You need it in Part 3 when you create an RKM stanza.
      4) Complete any other fields and click **Create**.
   d. After SKLM creates the device group, it prompts you to add devices and keys. Do not add any devices or keys. Instead, click **Close**. Keys are created in the next step.

6. Create master encryption keys for the device group.
   a. In the SKLM graphical user interface, in the **Key and Device Management** table, select the device group that you created in Step 5. In these instructions, the device group is named `GPFS_Tenant0001`.
   b. Click **Go to** > **Manage keys and services**.

c. In the management page for `GPFS_Tenant0001`, click **Add** > **Key**.

d. Enter the following information:

- The number of keys to be created.
- The three-letter prefix for key names. The key names are internal SKLM names and are not used for GPFS encryption.

e. Make a note of the UUID of the key, such as `KEY-326a1906-be46-4983-a63e-29f005fb3a15`. You need it in Part 3.

f. In the drop-down list at the bottom of the page, select **Hold new certificate requests pending my approval**.

Part 3: Configuring the remote key management (RKM) back end

To configure a remote key management (RKM) back end, you must create and initialize a client keystore and you must create an RKM stanza in the `RKM.conf` file on the IBM Spectrum Scale node:

1. On the IBM Spectrum Scale node that you are configuring for encryption, create the following subdirectory to contain the client keystore:

   `/var/mmfs/etc/RKMcerts`

2. Enter the following command to create the client credentials. The command is all on one line:

   ```
   mmgskkm gen --prefix /var/mmfs/etc/RKMcerts/SKLM --cname clientName
   --pwd-file passwordFile --fips fipsVal --nist nistVal --days validDays
   --keylen keyBits
   ```

   where:

   --prefix /var/mmfs/etc/RKMcerts/SKLM
   > Specifies the path and file name prefix of the client credential files that are generated.

   --cname clientName
   > Specifies the name of the client in the certificate.

   --pwd-file passwordFile
   > Specifies the path of a text file that contains the password for the client keystore. The password must be 1 - 20 characters in length.

   --fips fipsVal
   > Specifies the current FIPS 140-2 compliance mode of the IBM Spectrum Scale cluster. Valid values are `on` and `off`. To find the current mode, enter the following command:
   >
   > `mmlsconfig fips1402mode`

   --nist nistVal
   > Specifies the current NIST SP 800-131A compliance mode of the IBM Spectrum Scale cluster. Valid values are `on` and `off`. To find the current mode, enter the following command:
   >
   > `mmlsconfig nistCompliance`

   --days validDays
   > Specifies the number of days that the client certificate is valid.

   --keylen keyBits
   > Specifies the number of bits for the client private RSA key.

   The command creates three files that contain the private key, the public key, and the certificate for the client.

3. Enter the following command to create the client keystore and store the private key and the client certificate in it. The command is all on one line:

   ```
   mmgskkm store --cert /var/mmfs/etc/RKMcerts/SKLM.cert
   --priv /var/mmfs/etc/RKMcerts/SKLM.priv --label clientCertLabel
   --pwd-file passwordFile --out /var/mmfs/etc/RKMcerts/SKLM.p12
   --fips fipsVal --nist nistVal
   ```

where:

--cert /var/mmfs/etc/RKMcerts/SKLM.cert
> Specifies the path of the client certificate file. The path was specified in the `--prefix` parameter Step 2. The file suffix is `.cert`.

--priv /var/mmfs/etc/RKMcerts/SKLM.priv
> Specifies the path of the client private key file. The path was specified in the `--prefix` parameter in the Step 2. The file suffix is `.priv`.

--label clientCertLabel
> Specifies the label of the client certificate in the keystore.

--pwd-file passwordFile
> Specifies the path of a text file that contains the password for the client keystore. The password must be 1 - 20 characters in length.

--out /var/mmfs/etc/RKMcerts/SKLM.p12
> Specifies the path of the client keystore.

--fips fipsVal
> Specifies the current FIPS 140-2 compliance mode of the IBM Spectrum Scale cluster. Valid values are `on` and `off`. To find the current mode, enter the following command:
>
> ```
> mmlsconfig fips1402mode
> ```

--nist nistVal
> Specifies the current NIST SP 800-131A compliance mode of the IBM Spectrum Scale cluster. Valid values are `on` and `off`. To find the current mode, enter the following command:
>
> ```
> mmlsconfig nistCompliance
> ```

4. Copy the certificate files of the certificate chain, including the root certificate file, from the temporary directory to the directory that contains the client keystore. For more information, see Part 2. Rename each file with the same prefix, with a numeral that indicates the order of the certificate in the chain, and with the suffix `.cert`. Start the numbering with 0 for the root certificate. For example, if there are three files in the chain, and the prefix is `sklmChain`, rename the files as follows:

```
sklmChain0.cert
sklmChain1.cert
sklmChain2.cert
```

5. Enter the following command to verify the certificate chain. The command is all on one line:

```
openssl verify -CAfile /var/mmfs/etc/RKMcerts/sklmChain0.cert
-untrusted /var/mmfs/etc/RKMcerts/sklmChain1.cert
/var/mmfs/etc/RKMcerts/sklmChain2.cert
```

where:

-CAfile /var/mmfs/etc/RKMcerts/sklmChain0.cert
> Specifies the path of the root certificate file.

-untrusted /var/mmfs/etc/RKMcerts/sklmChain1.cert
> Specifies the path of the first intermediate certificate file. If no intermediate certificates are in the chain, omit this option. If multiple intermediate certificates are in the chain, combine all the intermediate certificates into a single file and pass the path of the combined file as the value to this option. In this example, there is a single intermediate certificate.

/var/mmfs/etc/RKMcerts/sklmChain2.cert
> Specifies the path of the endpoint certificate.

6. Enter the following command to store the certificate chain into the client keystore. The command is all on one line:

```
mmgskkm trust --prefix /var/mmfs/etc/RKMcerts/sklmChain
--pwd-file passwordFile --out /var/mmfs/etc/RKMcerts/SKLM.p12
--label labelChain --fips fipsVal --nist nistVal
```

where:

--prefix /var/mmfs/etc/RKMcerts/sklmChain

Specifies the path and the file name prefix of the files in the certificate chain. The **mmgsskm** command trusts all the files that have the specified prefix and a .cert suffix. For example, if there are three certificates in the chain and the prefix is /var/mmfs/etc/RKMcerts/ sklmChain, then the command trusts the following certificates:

/var/mmfs/etc/RKMcerts/sklmChain0.cert

/var/mmfs/etc/RKMcerts/sklmChain1.cert

/var/mmfs/etc/RKMcerts/sklmChain2.cert

--pwd-file passwordFile

Specifies the path of a text file that contains the password of the client keystore.

--out /var/mmfs/etc/RKMcerts/SKLM.p12

Specifies the path of the client keystore.

--label labelChain

Specifies the prefix of the label for the server certificate chain in the client keystore.

--fips fipsVal

Specifies the current FIPS 140-2 compliance mode of the IBM Spectrum Scale cluster. Valid values are `on` and `off`. To find the current mode, enter the following command:

```
mmlsconfig fips1402mode
```

--nist nistVal

Specifies the current NIST SP 800-131A compliance mode of the IBM Spectrum Scale cluster. Valid values are `on` and `off`. To find the current mode, enter the following command:

```
mmlsconfig nistCompliance
```

Important: The new keystore must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version of NFS or store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

7. Create an `RKM.conf` file and add a stanza to it that contains the information that is necessary to connect to the SKLM key server. The `RKM.conf` file must contain a stanza for each connection between a key client, an SKLM device group, and a key server.

   a. In a text editor, create a new text file with the following path and name:

      ```
      /var/mmfs/etc/RKM.conf
      ```

   b. Add a stanza with the following format:

      ```
      stanzaName {
          type = ISKLM
          kmipServerUri = tls://raclette.zurich.ibm.com:5696
          keyStore = /var/mmfs/etc/RKMcerts/SKLM.p12
          passphrase = a_password
          clientCertLabel = a_label
          tenantName = GPFS_Tenant0001
      }
      ```

      where the rows of the stanza have the following meaning:

      stanzaName

      A name (RKM ID) for the stanza. Make a note of the name: you need it in the next step.

      It is a good practice to use a format like the following one to ensure that the RKM ID is unique:

      `keyServerName_tenantName`

where *tenantName* is the name that you provide in the last line of stanza. For example, the RKM ID for the key server and key client in these instructions is: `raclette_GPFS_Tenant0001`.

type   Always ISKLM.

kmipServerUri
> The DNS name or IP address of the SKLM server and the KMIP SSL port. You can find this information on the main page of the SKLM graphic user interface. The default port is 5696.
>
> You can have multiple instances of this line, where each instance represents a different backup key server. The following example has the primary key server and two backup key servers:

```
stanzaName {
    type = ISKLM
    kmipServerUri = tls://raclette.zurich.ibm.com:5696
    kmipServerUri = tls://raclette.fondue.ibm.com:5696
    kmipServerUri = tls://raclette.fondue2.ibm.com:5696
    keyStore = /var/mmfs/etc/RKMcerts/SKLM.p12
    passphrase = a_password
    clientCertLabel = a_label
    tenantName = GPFS_Tenant0001
}
```

> If the GPFS daemon cannot get an encryption key from the primary key server, it tries the backup key servers in order.

keyStore
> The path and name of the client keystore.

passphrase
> The password of the client keystore and client certificate.

clientCertLabel
> The label of the client certificate in the client keystore.

tenantName
> The name of the SKLM device group. See "Part 1: Installing Security Key Lifecycle Manager" on page 606.

8. Set up an encryption policy on the node that you are configuring for encryption.

   a. Create a file management policy that instructs GPFS to do the encryption tasks that you want. The following is an example policy. It instructs IBM Spectrum Scale to encrypt all files in the file system with a file encryption key (FEK) and to wrap the FEK with a master encryption key (MEK):

```
RULE 'p1' SET POOL 'system' # one placement rule is required at all times
RULE 'Encrypt all files in file system with rule E1'
SET ENCRYPTION 'E1'
WHERE NAME LIKE '%'
RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-326a1906-be46-4983-a63e-29f005fb3a15:SKLM_srv')
```

   In the last line of the policy, the character string within single quotation marks (') is the key name. A *key name* is a compound of two parts in the following format:

   KeyID:RkmID

   where:

   KeyID Specifies the UUID of the key that you created in the SKLM graphic user interface in Part 2.

> *RkmID* Specifies the name of the RKM backend stanza that you created in the `/var/mmfs/etc/RKM.conf` file.

   b. Issue the **mmchpolicy** command to install the rule.

> **CAUTION:**
> **Installing a new policy with the mmchpolicy command removes all the statements in the previous policy. To add statements to an existing policy without deleting the previous contents, collect all policy statements for the file system into one file. Add the new statements to the file and install the contents of the file with the mmchpolicy command.**

9. Import the client certificate into the SKLM server:

   a. On the IBM Spectrum Scale node that you are configuring for encryption, send a KMIP request to SKLM. To send a KMIP request, try to create an encrypted file on the node. The attempt fails, but it causes SKLM to put the client certificate in a list of pending certificates in the SKLM key server. The attempt fails because SKLM does not yet trust the client certificate. See the following example:

```
# touch /gpfs0/test
touch: cannot touch `/gpfs0/test': Permission denied
# tail -n 2 /var/adm/ras/mmfs.log.latest
Thu Mar 20 14:00:55.029 2014: [E] Unable to open encrypted file: inode 46088,
Fileset fs1, File System gpfs0.
Thu Mar 20 14:00:55.030 2014: [E] Error: key
'KEY-326a1906-be46-4983-a63e-29f005fb3a15:SKLM_srv' could not be fetched (RKM
reported error -1004).
```

   b. In the graphical user interface of SKLM, on the main page, click **Pending client device communication certificates.**

   c. Find the client certificate in the list and click **View**.

   d. Carefully check that the certificate that you are importing matches the one created in the previous step, then click **Accept and Trust**.

   e. On the resulting screen, provide a name for the certificate and click **Accept and Trust** again.

   f. On the node that you are configuring for encryption, try to create an encrypted file as you did in Step (a). This time the command succeeds. Enter an **mmlsattr** command to list the encryption attributes of the new file:

```
# touch /gpfs0/test
# mmlsattr -n gpfs.Encryption /gpfs0/test
file name: /gpfs0/test
gpfs.Encryption: "EAGC????f?????????????? ??????w?^??>???????????? ?L4??
_-???V}f???X????,?G?<sH??0?)??M?????)?KEY-326a1906-be46-4983-a63e-29f005fb3a15?
sklmsrv?)?KEY-6aaa3451-6a0c-4f2e-9f30-d443ff2ac7db?RKMKMIP3?"
EncPar 'AES:256:XTS:FEK:HMACSHA512'
type: wrapped FEK WrpPar 'AES:KWRAP' CmbPar 'XORHMACSHA512'
KEY-326a1906-be46-4983-a63e-29f005fb3a15:sklmsrv
```

      From now on, the encryption policy rule causes each newly created file to be encrypted with a file encryption key (FEK) that is wrapped in a master encryption key (MEK). You created the key in a device group in the SKLM server and included its UUID as part of a key name in the security rule.

      Important: See the security note and the caution at the beginning of this topic, before Part 1.

Part 4: Enabling encryption on other nodes

1. To replicate an encryption configuration on another node, you must copy some configuration files from the configured node to the target node:

   a. Copy the `/var/mmfs/etc/RKM.conf` file to the same directory on the target node.

   b. Copy the keystore files that the RKM file references to the same directories on the target node. The suggested location for the keystore files on the configured node is `/var/mmfs/etc/RKMcerts/`.

2. To create a different encryption configuration on another node, follow the steps that are described in the preceding subtopics. Note the following design points:

   • On a single node, the following conditions are true:

– The `RKM.conf` file can contain multiple stanzas. Each stanza represents a connection between a key client and an SKLM device group.
– You can create multiple keystores.
- Across different nodes, the following conditions are true:
  – The contents of `RKM.conf` files can be different.
  – The contents of keystores can be different.
  – If an encryption policy succeeds on one node and fails on another in the same cluster, verify that the failing node has the correct client keystore and stanza.

Configuring encryption with SKLM v2.7

Learn to do tasks that are required for the Security Key Lifecycle Manager (SKLM) v2.7 server.

Learn to do the following tasks:

Resolving the UUID length problem for a single server

IBM Spectrum Scale allows a maximum length of 42 characters for the Universally Unique Identifier (UUID) of an encryption key. However, in SKLM v2.7 and later, SKLM generates UUIDs of up to 48 characters in length, including a 7 - 8 character Instance ID. You can work around this problem by configuring the SKLM server to use one-character instance IDs. After the configuration, the server generates UUIDs that have a maximum length of 42 characters. This method does not change existing UUIDs.

To configure a single SKLM server to use one-character Instance IDs, follow these steps.

1. Stop the SKLM server.
2. From the command line, change to the DB2/bin directory.

   Note: The location of the DB2/bin directory depends on the operating system:
   - On AIX, Linux, and similar operating systems, the directory is at the following location:

     `/opt/IBM/DB2SKLMV27/bin`
   - On Microsoft Windows, the directory is at the following location:

     `Drive:\Program Files\IBM\DB2SKLMV27\bin`

   If SKLM uses a preexisting DB2 installation, then the location of the `bin` directory might be different and might be on another system.
3. Start the DB2 command-line tool. The method depends on the operating system:
   - On AIX, Linux, and similar operating systems, enter the following command:

     `./db2`
   - On Microsoft Windows, enter the following command:

     `db2`
4. At the db2 command-line prompt, enter the following command to list the database directory:

   `list database directory`

   DB2 displays output like the following example:

   ```
   System Database Directory
    Number of entries in the directory = 1
   ```

```
Database 1 entry:

 Database alias                        = SKLMDB27
 Database name                         = SKLMDB27
 Local database directory              = /home/sklmdb27
 Database release level                = 14.00
 Comment                               =
 Directory entry type                  = Indirect
 Catalog database partition number     = 0
 Alternate server hostname             =
 Alternate server port number          =
```

Make a note of the database name.

5. Enter the following command to connect to the SKLM database:

   ```
   connect to database user userName using password
   ```

   where:

   database
   > Specifies the database name from the previous step.

   userName
   > Specifies the SKLM DB2 user name that you set during SKLM installation. The default value is `sklmdb27`.

   password
   > Specifies the SKLM DB2 password that you set during SKLM installation.

6. Enter the following command to change the SKLM instance ID. The command is on one line:

   ```
   update KMT_CFGT_INSTDETAILS set INSTANCEID='1' where INSTANCEID in
   (select INSTANCEID from KMT_CFGT_INSTDETAILS)
   ```

   where 1 is the one-character Instance ID that you want to set. DB2 displays output like the following example:

   ```
   DB20000I  The SQL command completed successfully.
   ```

7. Enter the following command to commit the change:

   ```
   commit
   ```

   DB2 displays output like the following example:

   ```
   DB20000I  The SQL command completed successfully.
   ```

8. Enter the following command to close the DB2 command-line tool:

   ```
   quit
   ```

9. Start the SKLM system.

The server now generates UUIDs that have a maximum length of 42 characters.

Resolving the UUID length problem in a high-availability cluster

IBM Spectrum Scale allows a maximum length of 42 characters for the Universally Unique Identifier (UUID) of an encryption key. However, in SKLM v2.7 and later, SKLM generates UUIDs of up to 48 characters in length, including a 7 - 8 character Instance ID. You can work around this problem by configuring the SKLM server to use one-character instance IDs. After the configuration, the server generates UUIDs that have a maximum length of 42 characters. This method does not change existing UUIDs.

To configure the SKLM servers in a high-availability (HA) cluster, follow these steps.

1. Stop the SKLM server on all the nodes of the cluster.
2. For each node in the cluster, follow the steps in the preceding subtopic "Resolving the UUID length problem for a single server". Set the Instance ID to a separate value in each node.

3. Start the SKLM server on all nodes in the cluster.

The servers now generate UUIDs that have a maximum length of 42 characters.

Changing the REST port: Simplified setup
Changing the REST port affects only the simplified setup, not the regular setup.

In SKLM v2.6, the default Representational State Transfer (REST) port is 9080. In SKLM v2.7 and later, the default REST port is 443.

To change the REST port number that IBM Spectrum Scale uses to connect to SKLM to 443, enter the following command:
```
mmkeyserv server add --port 443
```

For more information, see "Simplified setup: Using SKLM with a self-signed certificate" on page 568.

Upgrading SKLM to v2.7: Regular setup
Upgrading SKLM to v2.7 from v2.5 or v2.6 has no adverse impacts on the regular setup, other than the UUID length problem that is described earlier in the topic.

Upgrading SKLM to v2.7: Simplified setup
After you upgrade SKLM to v2.7 from v2.5 or v2.6, you can no longer run the following commands to add tenants, register clients, and create keys:
```
mmkeyserv tenant add
mmkeyserv client register
mmkeyserv key create
```

The reason is that these commands are configured with the SKLM v2.5 or v2.6 default REST port, whose value is different in v2.7. For more information, see the previous subtopic on changing the REST port.

To resolve this problem, follow these steps:
1. Unregister each key client from its tenant by running the **mmkeyserv deregister** command, as in the following example:
   ```
   mmkeyserv deregister ClientName --tenant TenantName --server-pwd PasswordFile
   ```
2. Delete each tenant by running the **mmkeyserv tenant delete** command, as in the following example:
   ```
   mmkeyserv tenant delete TenantName --server ServerName --server-pwd
   ```
3. Delete the server with the **mmkeyserv server delete** command, as in the following example:
   ```
   mmkeyserve server delete ServerName
   ```
4. Upgrade SKLM to v2.7.
5. Add the server, tenants, and key clients as described in "Simplified setup: Using SKLM with a self-signed certificate" on page 568.

   Note: You do not have to delete and re-create encryption keys. You can use the same encryption keys in SKLM v7 that you configured in SKLM v2.5 or v2.6.

Configuring encryption with the Vormetric DSM key server

Setting up an encryption environment with Vormetric Data Security Manager (DSM) key server requires IBM Spectrum Scale Advanced Edition V4.2.1 or later and Vormetric DSM V5.2.3 or later.

You should be aware of the following items:

The IBM Spectrum Scale node that you are configuring for encryption must have direct network access to the system where the key server is installed.

SECURITY NOTE: The contents of the following files are security-sensitive:
- The `RKM.conf` file:
  - For the simplified setup: `/var/mmfs/ssl/keyServ/RKM.conf`
  - For the regular setup and the Vormetric DSM setup: `/var/mmfs/etc/RKM.conf`
- The directory for the client keystore:
  - For the simplified setup: `/var/mmfs/ssl/keyServ`
  - For the regular setup and the Vormetric DSM setup: `/var/mmfs/etc/RKMcerts`

IBM Spectrum Scale reads the contents of security-sensitive files only if the following conditions are met:
- They are regular files that are owned by the root user.
- They are in the root group.
- They are readable and writable only by the user.

See the permission bits in the following examples:
- For the simplified setup:

```
-rw-------. 1 root root 2454 Mar 20 10:32 /var/mmfs/ssl/keyServ/RKM.conf
drw-------. 2 root root 4096 Mar 20 11:15 /var/mmfs/ssl/keyServ/
-rw-------. 1 root root 3988 Mar 20 11:15 /var/mmfs/ssl/keyServ/keystore_name.p12
```

  Note: In the simplified setup, the **mmkeyserv** command sets the permission bits automatically.
- For the regular setup and the Vormetric DSM setup:

```
-rw-------. 1 root root 2446 Mar 20 12:15 /var/mmfs/etc/RKM.conf
drw-------. 2 root root 4096 Mar 20 13:47 /var/mmfs/etc/RKMcerts
-rw-------. 1 root root 3988 Mar 20 13:47 /var/mmfs/etc/RKMcerts/keystore_name.p12
```

CAUTION:
It is a good practice to take the following precautions:
- **Ensure that the passphrase for the client certificate file is not leaked through other means, such as the shell history.**
- **Take appropriate precautions to ensure that the security-sensitive files are not lost or corrupted. IBM Spectrum Scale does not manage or replicate the files.**

Important: The client keystore must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version of NFS or store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

See the following subtopics for instructions:

Part 1: Creating credentials for the key client
- Some of the commands in the following instructions require you to specify values for the following two parameters:

  --fips Specifies whether the key client complies with the requirements of FIPS 140-2.

  --nist Specifies whether security transport for the key client complies with the NIST SP800-131A recommendations.

  For both parameters, follow these guidelines:
  - If the key client complies, set the parameter to **on**; otherwise, set the parameter to **off**.

- Specify the same setting for each parameter as the setting in the IBM Spectrum Scale cluster. To display these settings, enter the following two commands:

```
mmlsconfig nistCompliance
mmlsconfig FIPS1402mode
```

Follow these steps:

1. On the IBM Spectrum Scale node that you are configuring for encryption, run the **mmgskkm** command to create the client credentials. Enter the following command on one line:

```
/usr/lpp/mmfs/bin/mmgskkm gen --prefix prefix --cname cname --pwd pwd --fips fips --nist nist
    --days valid_days --keylen keylen
```

where:

--prefix *prefix*

Specifies the path and file name prefix of the directory where the output files are generated. For example, if you want directory /var/mmfs/etc/RKMcerts to contain the output files, and you want the output files to have the prefix kcVormetric, you can specify the parameter as follows:

```
--prefix /var/mmfs/etc/RKMcerts/kcVormetric
```

--cname *cname*

Specifies the name of the IBM Spectrum Scale key client. Valid characters are alphanumeric characters, hyphen (-), and period (.). The name can be up to 54 characters long. In Vormetric DSM, names are not case-sensitive, so the use of uppercase letters is not recommended. For more information, see the Vormetric DSM documentation.

--pwd *pwd*

Specifies the password for the private key that this command creates.

--fips *fips*

Specifies whether the key client complies with FIPS 140-2. Specify **on** or **off**.

--nist *nist*

Specifies whether the key client complies with NIST SP800-131a. Specify **on** or **off**.

--validdays *validdays*

Specifies the number of days that the client certificate is valid.

--keylen *keylen*

Specifies the length in bits of the RSA key that is generated.

In the following example, the current directory is the output directory. Enter the command on one line:

```
/usr/lpp/mmfs/bin/mmgskkm gen --prefix kcVormetric --cname kcVormetric --pwd pwpkVormetric
    --fips off --nist on --days 180 --keylen 2048
```

The output files are a client certificate, a private key, and a public key. For example:

```
kcVormetric.cert
kcVormetric.priv
kcVormetric.pub
```

2. Run the **mmgskkm** command again to create a PKCS#12 keystore and to store the certificate and private key of the client in it. Enter the following command on one line:

```
/usr/lpp/mmfs/bin/mmgskkm store --cert certFile --priv privFile --label label --pwd pwd --out keystore
```

where:

--cert *certFile*

Specifies the client certificate file that you created in Step 1.

--priv *privFile*
> Species the private key file that you created in Step 1.

--label *label*
> Specifies the label under which the private key is stored in the keystore.

--pwd *pwd*
> Specifies the password of the keystore. You can use the same password that you specified for the private key in Step 1.

--out *keystore*
> The file name of the keystore.

In the following example, the current directory contains the client credentials from Step 1. The command is entered on one line:

```
mmgskkm store --cert kcVormetric.cert --priv kcVormetric.priv --label lapkVormetric
    --pwd pwpkVormetric --out ksVormetric.keystore
```

The output file is a keystore that contains the client credentials of the key client:

```
ksVormetric.keystore
```

Important: The keystore must be record-locked when the GPFS daemon starts. If the keystore files are stored on an NFS mount, the encryption initialization process can hang. The cause is a bug that affects the way NFS handles record locking. If you encounter this problem, upgrade your version of NFS or store your keystore file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the keystore files.

3. Retrieve the certificate chain of the Vormetric DSM server.

Note: Before you can do this next step, you must install the Vormetric DSM server, set up the DSM networking configuration, and set up the server certificate. If you do not, then you might not be able to connect to the DSM server or you might retrieve an invalid, default certificate chain.

Note: DSM does not support the use of imported server certificate chains for the TLS communication on the KMIP port. You must create and use a server certificate chain signed by the DSM internal certificate authority (CA).

Enter the following command on one line:

```
/usr/lpp/mmfs/bin/mmsklmconfig restcert --host host --port port --prefix prefix --keystore keystore
    --keypass keypass --fips fips --nist nist
```

where:

--host *host*
> Specifies the name or IP address of the remote system where the Vormetric DSM server is running.

--port *port*
> Specifies the port on the remote system for communicating with the Vormetric DSM server (default 8445).

--prefix *prefix*
> Specifies the path and file name prefix of the directory where the files in the certificate chain are stored. For example, if you want to store the certificate chain in the directory /var/mmfs/etc/RKMcerts, and you want the certificate files to have the prefix DSMServer, you can specify the parameter as follows:
>
> --prefix /var/mmfs/etc/RKMcerts/DSMServer

--keystore *keystore*
> Specifies the path and file name of the client keystore that you created in Step 2.

--keypass *keypass*

> Specifies a text file that contains the password of the client keystore as the first line. You must create this text file. Store the password that you provided in Step 2.

--fips *fips*

> Specifies whether the key client complies with FIPS 140-2. Specify **on** or **off**.

--nist *nist*

> Specifies whether the key client complies with NIST SP800-131a. Specify **on** or **off**.

In the following example, the current directory contains the client keystore that was created in Step 2. Enter the command on one line:

```
/usr/lpp/mmfs/bin/mmsklmconfig restcert --host hostVormetric --port 8445 --prefix DSM
    --keystore ksVormetric.keystore --keypass keypass --fips off --nist on
```

The command connects to the DSM server, retrieves the server certificate chain, and stores each certificate into a separate local file in Base64-encoded DER format. Each file name has the format *prefixN*.**cert**, where *prefix* is the prefix that you specified in the command and *N* is a digit that begins at 0 and increases by 1 for each certificate in the chain, as in the following example:

> DSM0.cert
>
> DSM1.cert

4. Verify that the SHA-256 fingerprint in each retrieved certificate matches the fingerprint of the DSM server:

   a. To display the details of each certificate, enter the following sequence at the client command line, where *prefix* is the prefix that you provided in Step 3:

   ```
   for c in prefix*.cert; do /usr/lpp/mmfs/bin/mmgskkm print --cert $c; done
   ```

   b. Log in to the graphical user interface of the DSM server and display its SHA-256 fingerprint.

   c. Verify that the fingerprints in the certificates match the fingerprint in the DSM server.

5. Add the certificates to the PKCS#12 keystore of the key client as trusted certificates. Enter the following command on one line:

```
/usr/lpp/mmfs/bin/mmgskkm trust --prefix prefix --pwd pwd --out keystore --label serverLabel
    --fips fips --nist nist
```

where:

--prefix *prefix*

> Specifies the prefix that you specified in Step 3.

--pwd *pwd*

> Specifies the password of the client keystore, which you provided in Step 3.

--out *keystore*

> Specifies the path name of the keystore of the key client.

--label *serverLabel*

> Specifies the label under which the server certificate chain is stored in the client keystore.

--fips *fips*

> Specifies whether the key client complies with FIPS 140-2. Specify **on** or **off**.

--nist *nist*

> Specifies whether the key client complies with NIST SP800-131a. Specify **on** or **off**.

In the following example, the current directory contains the client keystore and the certificate chain. Enter the following command on one line:

```
/usr/lpp/mmfs/bin/mmgskkm trust --prefix DSM --pwd pwpkVormetric --out ksVormetric.keystore
    --label laccVormetric --fips off --nist on
```

The keystore of the key client contains the following items:

- Client credentials

- The certificate chain of the Vormetric DSM key server as trusted certificates

Part 2: Configuring the Vormetric DSM key server

The following instructions describe how to configure the Vormetric Data Security Manager (DSM) key server to communicate with an IBM Spectrum Scale key client.

In DSM, a *host* is a system to which DSM provides security services. In these instructions, the host is the IBM Spectrum Scale node that you are configuring for encryption. A DSM *domain* is an administrative group of one or more hosts. In these instructions, the domain contains the single IBM Spectrum Scale node. For more complex configurations, see the DSM product documentation.

1. Install a Key Management Interoperability Protocol (KMIP)-enabled license in DSM.

   Important: You must complete this step before you create a DSM domain. For security reasons, you cannot create a KMIP-enabled domain in DSM until you install a KMIP-enabled license. For example, you cannot create a regular domain, install a KMIP-enabled license, and then convert the domain to a KMIP-enabled domain.

   a. On the DSM Management Console, click **System** > **License**.

   b. Select a KMIP-enabled license that you obtained from DSM.

   c. Click **Upload License File**.

   The license is installed.

2. Create a DSM domain.

   a. On the DSM Management Console, click **Domains** > **Manage Domains**.

   b. Follow the instructions to create a domain. Make sure that you configure the domain as **KMIP Supported**.

3. Create a Domain and Security Administrator for the new domain.

   Note: In these instructions, a single Domain and Security Administrator is created who combines the responsibilities of administering the domain and controlling its security. For security reasons, you might want to create a Domain Administrator and a Security Administrator as separate roles. For more information, see the DSM documentation.

   a. Log in as the DSM System Administrator. On the Management Console, click **Administrators**.

   b. On the Administrators page, click **Add**.

   c. In the Add Administrator window, complete all the input fields except the **RSA User ID** field. In the **User Type** field, click **Domain and Security Administrator**.

   Note: The passwords are temporary. The new administrator must enter a new password on the first login to the DSM Management Console.

   d. Click **OK**.

   e. Limit the scope of the administrator's control to the domain that you created in Step 2.

4. Add a host to the domain.

   a. Log in as the new administrator:

      1) Enter a password when prompted.

      2) Select **I am a local domain administrator**.

      3) Enter or select the domain name from Step 2.

   b. On the Management Console, click **Hosts** > **Hosts**.

   c. On the Hosts screen, click **Add** to add a KMIP host. Set the **Host Name** to the name that you specified for the key client (the value for the **cname** parameter) when you created the client credentials in Part 1. In these instructions, the key client name is kcVormetric.

d. In the list of hosts, select the host that you created in the previous step. Click **Import KMIP Cert**. If no **Import KMIP Cert** button is displayed, verify that the DSM license is KMIP-enabled and that you created the domain after you installed the KMIP-enabled license.

e. In the window that opens, go through the directories of the IBM Spectrum Scale node to the directory that contains the client certificate file. Select the certificate file.

5. Create one or more keys for the client to use as master encryption keys (MEKs).

a. From the DSM Management Console, click **Keys** > **Key Templates.** Follow the instructions to create a key template. Select **AES256** as the key algorithm.

b. Create a key from the template. Specify a name for the key and then select the template.

c. Make a note of the UUID of the key. You need it in Part 3.

Part 3: Configuring the IBM Spectrum Scale node

1. Create an `RKM.conf` file and add a remote key management (RKM) stanza to it that contains the information that is necessary to communicate with the Vormetric DSM key server.

a. On the IBM Spectrum Scale node, create a text file with the following path and name:

`/var/mmfs/etc/RKM.conf`

b. Add a stanza with the following format:

```
stanzaName {
    type = KMIP
    kmipServerUri = tls://raclette.zurich.ibm.com:5696
    keyStore = /var/mmfs/etc/RKMcerts/ksVormetricDMS.p12
    passphrase = a_password
    clientCertLabel = a_label
}
```

where the rows of the stanza have the following meanings:

stanzaName
   A name (RKM ID) for the stanza. Make a note of the name: you need it in the next step.

   It is a good practice to use a format like the following one to ensure that the RKM ID is unique:
   `keyServerName_keyClientName`

   where *keyClientName* is the key client name from Part 1, Step 1. For example, the RKM ID for the key server and key client in these instructions is: `raclette_kcVormetric`.

type   Always KMIP for the Vormetric DSM server.

kmipServerUri
   The DNS name or IP address of the DSM server and the DSM SSL port. Multiple kmipServerUri entries may be added for high availability (HA), but note that the DSM servers must then be configured in an active-active setup. In the regular DSM HA setup, the passive failover nodes do not serve keys over KMIP. For more information, consult the Vormetric DSM documentation.

keyStore
   The path and name of the client keystore from Part 1.

passphrase
   The password of the client keystore and client certificate from Part 1.

clientCertLabel
   The label of the client certificate in the client keystore from Part 1.

2. Set up an encryption policy on the node that you are configuring for encryption.

a. Create a policy that instructs GPFS to do the encryption tasks that you want. The following policy is an example policy. It instructs IBM Spectrum Scale to encrypt all files in the file system with a file encryption key (FEK) and to wrap the FEK with a master encryption key (MEK):

```
RULE 'p1' SET POOL 'system' # one placement rule is required at all times
RULE 'Encrypt all files in file system with rule E1'
SET ENCRYPTION 'E1'
WHERE NAME LIKE '%'
RULE 'simpleEncRule' ENCRYPTION 'E1' IS
ALGO 'DEFAULTNISTSP800131A'
KEYS('01-10:raclette_kcVormetric')
```

In the last line, the character string within single quotation marks (') is the key name. A *key name* is a compound of two parts in the following format:

```
KeyID:RkmID
```

where:

KeyID

Specifies the UUID of the master encryption key that you created in the DSM Management Console in Part 2.

RkmID

Specifies the name of the RKM stanza that you created in the `/var/mmfs/etc/RKM.conf` file in Step 1.

b. Install the policy rule with the **mmchpolicy** command.

CAUTION:

Installing a new policy with the mmchpolicy command removes all the statements in the previous policy. To add statements to an existing policy without deleting the previous contents, collect all policy statements for the file system into one file. Add the new statements to the file and install the contents of the file with the mmchpolicy command.

From now on, the encryption policy rule causes each newly created file to be encrypted with a file encryption key (FEK) that is wrapped in a master encryption key (MEK).

Part 4: Enabling encryption on other nodes

1. To replicate an encryption configuration on another node, you must copy some configuration files from the configured node to the target node:

    a. Copy the `/var/mmfs/etc/RKM.conf` file to the same directory on the target node.

    b. Copy the keystore files that the RKM file references to the same directories on the target node. The recommended location for the keystore files on the configured node is `/var/mmfs/etc/RKMcerts/`.

2. To create a different encryption configuration on another node, follow the steps that are described in the preceding subtopics. Note the following design points:

    • On a single node:

        – The `RKM.conf` file can contain multiple stanzas. Each stanza represents a connection between a key client and a DSM host.

        – You can create multiple keystores.

    • Across different nodes:

        – The contents of `RKM.conf` files can be different.

        – The contents of keystores can be different.

        – If an encryption policy succeeds on one node and fails on another in the same cluster, verify that the failing node has the correct client keystore and stanza.

Renewing client and server certificates

This topic describes IBM Spectrum Scale client and server certificates.

During encryption, the GPFS daemon acts as a key client and requests master encryption keys (MEKs) from a Remote Key Management (RKM) server. The supported RKM servers are IBM Security Key Lifecycle Manager (SKLM) and Vormetric Data Security Manager (DSM).

The following table shows the default lifetimes of client and server certificates:

Table 55. Default lifetimes of key server and key client certificates

Item	Type of certificate	Default lifetime
IBM Spectrum Scale	Client	3 years
IBM Security Key Lifecycle Manager (SKLM)	Server	3 years
Vormetric Data Security Manager (DSM)	Server	10 years

Certificate expiration errors

Learn how IBM Spectrum Scale checks for expired RKM server and key client certificates and what actions it takes when it finds an expired certificate.

Expired certificate for an RKM server

When the certificate of an RKM server expires, IBM Spectrum Scale can no longer retrieve master encryption keys (MEKs) from the server. The result is that attempts to create, open, read, or write encrypted files fail with an "Operation not permitted" error. Each time that an error occurs, IBM Spectrum Scale writes error messages like the following ones to the /var/adm/ras/mmfs.log.latest log file:

```
[W] The key server sklm1 (port 5696) had a failure and will be
    quarantined for 1 minute(s).
[E] Unable to create encrypted file testfile.enc (inode 21260,
    fileset 0, file system gpfs1).
[E] Key 'KEY-uuid:sklm1' could not be fetched. Bad certificate.
```

Expired certificate for a key client

IBM Spectrum Scale checks the status of a key client certificate each time it loads a keystore. It loads a keystore whenever a file system is mounted, a new policy is applied, or an RKM.conf configuration file is explicitly loaded with the tsloadikm run command.

When IBM Spectrum Scale detects an expired client certificate, it writes one or more of the following error messages to the /var/adm/ras/mmfs.log.latest log file or to the console or to both, depending on the action that you took when the problem occurred:

```
[E] Error while validating policy 'policy.enc': rc=778:
While parsing file '/var/mmfs/etc/RKM.conf':
[E] Incorrect client certificate label 'GPFSlabel' for backend 'sklm2'.
[E] Error while validating policy 'policy.enc': rc=778:
While parsing file '/var/mmfs/etc/RKM.conf':
[E] Certificate with label 'GPFSlabel' for backend 'sklm2' has expired.
```

Renewing expired server certificates

Follow these instructions to renew expired server certificates for the simplified setup, the regular setup, and certificate chains.

See the following topics for detailed instructions:

| **Creating a server certificate:**
| The steps for creating a new server certificate to replace one that is expired are similar to the steps for
| creating an initial server certificate. Follow the instructions in the documentation of your Remote Key
| Manager (RKM), which must be one of the following products:
| • IBM Security Key Lifecycle Manager (SKLM)
| • Vormetric Data Security Manager (DSM)

| For more information, see the section *Establishing an encryption-enabled environment* in the *IBM Spectrum*
| *Scale: Administration Guide*.

| **Simplified setup: Trusting a new self-signed SKLM server certificate:**
| These instructions assume that you are using the simplified setup method and that you have created a
| self-signed SKLM server certificate.
| 1. Find the name of the key server object that needs to be updated. To display a list of the available key
| server objects, issue the following command on the IBM Spectrum Scale command line:
| `mmkeyserv server show`
| 2. Issue the following command to update the server certificate of the key server object:
| `mmkeyserv server update <serverName>`

| where `<serverName>` is the name of the key server object that you want to update.
| 3. Enter the SKLMAdmin administrator password when prompted.
| 4. Enter `yes` to trust the SKLM REST certificate.

| The key server object is updated with the self-signed server certificate.

| **Simplified setup: Trusting a new SKLM WebSphere Application Server certificate:**
| These instructions assume that you are using the simplified setup method with IBM WebSphere
| Application Server and SKLM.
| 1. The simplified setup communicates with SKLM on both the KMIP port and the REST administration
| port. On the REST port, the server certificate is the one that is configured in WebSphere Application
| Server. SKLM runs on top of WebSphere Application Server.
| 2. Find the name of the IBM Spectrum Scale key server object that is associated with SKLM on the REST
| port. To see a list of key server objects, issue the following command:
| `mmkeyserv server show`
| 3. Issue the following command to update the key server object with the new WebSphere Application
| Server certificate:
| `mmkeyserv server update <serverName>`

| where `<serverName>` is the name of the key server object that you want to update.
| 4. Enter the SKLMAdmin administrator password when prompted.
| 5. Enter `yes` to trust the SKLM REST certificate.

| The IBM Spectrum Scale client now trusts the new SKLM WebSphere Application Server certificate.

| **Regular setup: Trusting a new self-signed SKLM server certificate:**
| Follow these instructions if you are using IBM Spectrum Scale v4.1.1 or later. These instructions assume
| that you are using SKLM and the regular setup method and that you have created a self-signed SKLM
| server certificate.
| 1. Get information about the key client from the `/var/mmfs/etc/RKM.conf` file:
| a. Open the file and find the RKM stanza for the key client that you want to configure.
| b. Make a note of the following information from the RKM stanza:

- The password for the client keystore and client certificate, which is specified by the **passphrase** term. You need this information for Step 2.
- The path and file name of the client keystore, which is specified by the **keyStore** term. You need this information for Step 3.

2. Store the client keystore password from Step 1 into a text file, such as /root/keystore.pwd, that is accessible only by the root user.

3. Issue the **mmsklmconfig** command to retrieve the new self-signed SKLM server certificate. This command is available in IBM Spectrum Scale v4.2.1 and later. The command connects to the KMIP port, waits for the TLS handshake, and retrieves the certificate that the server presents.

```
mmsklmconfig restcert --host <sklmhost> --port <kmipport>
--prefix <sklmChain> --keystore <rkmKeystore>
--keypass <rkmPassfile> --fips <fips> --nist <nist>
```

where:

--host *<sklmhost>*
    Is the IP address or host name of the RKM server.

--port *<kmipport>*
    Is the KMIP port number of the SKLM server. The default value is 5696.

--prefix *<sklmChain>*
    Is the path and file name prefix where the server certificate files are to be stored.

--keystore *<rkmKeystore>*
    Is the path and file name of the client keystore from Step 1.

--keypass *<rkmPassfile>*
    Is the path and file name of the keystore password file from Step 2.

--fips *<fips>*
    Indicates whether the IBM Spectrum Scale cluster is using FIPS 140-2-compliant cryptographic modules. Valid values are **on** or **off**. Enter the following command to determine the state:
    ```
    mmlsconfig FIPS1402mode
    ```

--nist *<nist>*
    Indicates whether the IBM Spectrum Scale cluster is using encryption that is in compliance with NIST SP800-131A recommendations. Valid values are **on** or **off**. Enter the following command to determine the state:
    ```
    mmlsconfig nistCompliance
    ```

4. Optionally, print the contents of the retrieved server certificate file and verify that the information matches the information in the new server certificate on the RKM server. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later. Issue the following command:
```
mmgskkm print --cert sklmChain0.cert
```

where *sklmChain* is the path and file name prefix of the certificate files. You specified this prefix in Step 3.

5. Issue the following command to add the retrieved server certificate to the client keystore: The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later.
```
mmgskkm trust --prefix <sklmChain> --out <rkmKeystore> --pwd-file <rkmPassfile>
--label <serverLabel>
```

where:

--prefix *<sklmChain>*
    Is the path and file name prefix of the server certificate files. You specified this prefix in Step 3.

--out *<rkmKeystore>*
> Is the path and file name of the client keystore from Step 1.

--pwd-file *<rkmPassfile>*
> Is the path and file name of the client keystore password file that you created in Step 2.

--label *<serverLabel>*
> Is the label under which to store the server certificate in the client keystore.
>
> > **Note:** The label must be unique in the keystore. In particular, it cannot be the label of the expired server certificate from the SKLM key server.

6. Copy the updated client keystore file to all the nodes in the IBM Spectrum Scale cluster.
7. Reload the new client keystore by one of the following methods:
   - On any administration node in the cluster, run the **mmchpolicy** command to refresh the current policy rules. You do not need to repeat this action on other nodes in the cluster.
   - On each node of the cluster, unmount and mount the file system.
   - In IBM Spectrum Scale v4.2.1 and later, issue the following command on each node of the cluster:

     `/usr/lpp/mmfs/bin/tsloadikm run`

The IBM Spectrum Scale client now trusts the new self-signed SKLM server certificate.

Trusting a new endpoint server certificate in a server certificate chain:
These instructions assume that the certificate chain consists of more than one certificate and that it includes a root certificate that is signed by a certificate authority (CA), optional intermediate certificates, and an endpoint certificate.
1. If only the endpoint certificate expired and was renewed, you do not need to take any further action on the client side. This situation occurs, for example, in Vormetric DSM when you renew an endpoint certificate by running the **gencert** command.
2. If an intermediate certificate or the root certificate expired and was renewed, follow the instructions in one of the following two subtopics:

   "Regular setup: Trusting a new SKLM server certificate chain"

   "Trusting a new Vormetric DSM server certificate chain" on page 629

Regular setup: Trusting a new SKLM server certificate chain:
These instructions assume that you are using SKLM and the regular setup method and that you have a certificate chain from a Certificate Authority (CA) in which you have renewed an intermediate certificate or the root certificate. For information about obtaining a certificate chain from a CA, see the subtopic "Part 2: Configuring SKLM" in "Regular setup: Using SKLM with a certificate chain" on page 604.
1. Get the path and password of the keystore file of the key client that you are configuring:
   a. Open the `/var/mmfs/etc/RKM.conf` file and find the RKM stanza of the key client .
   b. Make a note of the following items:
      - The password for the client keystore and client certificate, which is specified by the **passphrase** term. You need this information for Step 2.
      - The path and file name of the client keystore, which is specified by the **keyStore** term. You need this information for Step 5.
2. Store the client keystore password from Step 1 into a text file, such as `/root/keystore.pwd`, that is accessible only by the root user.
3. Set up the files in the certificate chain:
   a. Copy the files for the new server certificate chain into the same directory in which the `keystore.pwd` file is located.
   b. Rename each file with the same path and file name prefix, followed by an increasing integer value that indicates the order of the certificate in the chain, followed by the suffix `.cert`. Start the

numbering with 0 for the root certificate. The following example shows the renamed certificate files for a server certificate chain that consists of a root CA certificate, one intermediate certificate, and an endpoint certificate:

    /root/sklmChain0.cert (Root certificate)

    /root/sklmChain1.cert (Intermediate certificate)

    /root/sklmChain2.cert (Endpoint certificate)

4. Optionally, you can verify the server certificate chain by issuing the **openssl verify** command. The command has the following usage:

```
openssl verify -CAfile <rootCaCert> [-untrusted <intermediateCaCerts>] <endpointCert>
```

where:

-CAfile *<rootCaCert>*
> Specifies the root certificate file.

-untrusted *<intermediateCaCerts>*
> Specifies the file that contains the intermediate certificates. If the chain has more than one intermediate certificate, you must combine them into a single file. If the chain has no intermediate certificates, omit this parameter.

<endpointCert>
> Specifies the endpoint certificate file.

For example, if your server certificate chain consists of the three sample files that are listed in Step 3, issue the following command:

```
openssl verify -CAfile /root/sklmChain0.cert -untrusted /root/sklmChain1.cert /root/sklmChain2.cert
```

5. Issue the following command to add the new SKLM server certificate chain to the keystore. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later.

```
mmgskkm trust --prefix <sklmChain> --out <keystore> --pwd-file <pwd-file>
--label <serverLabel>
```

where:

--prefix *<sklmChain>*
> Is the path and file name prefix of the certificate chain files that you set up in Step 3, such as /root/sklmChain.

--out *<keystore>*
> Is the path and file name of the client keystore from Step 1.

--pwd-file *<pwd-file>*
> Is the path and file name prefix of the keystore password file that you created in Step 2.

--label *<serverLabel>*
> Is the label under which to store the server certificate in the client keystore.

> **Note:** The label must be unique in the keystore. Also, it cannot be the label of the expired server certificate from the SKLM key server.

6. Copy the updated client keystore to all nodes in the IBM Spectrum Scale cluster.
7. Reload the new client keystore by one of the following methods:
   - On any administration node in the cluster, run the **mmchpolicy** command to refresh the current policy rules. You do not need to repeat this action on other nodes in the cluster.
   - On each node of the cluster, unmount and mount the file system.
   - In IBM Spectrum Scale v4.2.1 and later, issue the following command on each node of the cluster:
     ```
     /usr/lpp/mmfs/bin/tsloadikm run
     ```

The IBM Spectrum Scale client now trusts the new SKLM server certificate chain.

Trusting a new Vormetric DSM server certificate chain:
These instructions assume that you are using Vormetric DSM and that you have a DSM certificate chain that you have renewed by running the `security genca` command.

1. Get the path and password of the keystore file of the key client that you are configuring:

   a. Open the `/var/mmfs/etc/RKM.conf` file and find the RKM stanza of the key client.

   b. Make a note of the following items:

      • The password for the client keystore and client certificate, which is specified by the **passphrase** term. You need this information for Step 2.

      • The path and file name of the client keystore, which is specified by the **keyStore** term. You need this information for Step 5.

2. Store the client keystore password from Step 1 into a text file, such as `/root/keystore.pwd`, that is accessible only by the root user.

3. Issue the **mmsklmconfig** command to retrieve the new self-signed DSM server certificate chain. This command is available in IBM Spectrum Scale v4.2.1 and later. The command connects to the KMIP port, waits for the TLS handshake, and retrieves the certificate that the server presents.

```
mmsklmconfig restcert --host <dsmhost> --port <dsmport>
--prefix <dsmChain> --keystore <rkmKeystore>
--keypass <rkmPassfile> --fips <fips> --nist <nist>
```

   where:

   --host *<dsmhost>*
   Is the IP address or host name of the DSM server.

   --port *<dsmport>*
   Is the port number of the DSM web GUI. The default value is 8445.

   --prefix *<sklmChain>*
   Is the path and file name prefix where the server certificate files are to be stored.

   --keystore *<rkmKeystore>*
   Is the path and file name of the client keystore from Step 1.

   --keypass *<rkmPassfile>*
   Is the path and file name of the keystore password file from Step 2.

   --fips *<fips>*
   Indicates whether the IBM Spectrum Scale cluster is using FIPS 140-2-compliant cryptographic modules. Valid values are **on** or **off**. Enter the following command to determine the state:
   ```
   mmlsconfig FIPS1402mode
   ```

   --nist *<nist>*
   Indicates whether the IBM Spectrum Scale cluster is using encryption that is in compliance with NIST SP800-131A recommendations. Valid values are **on** or **off**. Enter the following command to determine the state:
   ```
   mmlsconfig nistCompliance
   ```

   DSM server certificate chain: The DSM server certificate chain typically consists of two certificates, a DSM internal root CA certificate and an endpoint certificate. The names of certificate files that you retrieve in this step have the following format: the path and file name prefix that you specify in the **--prefix** parameter, followed by a 0 for the root certificate or a 1 for the endpoint certificate, followed by the suffix `.cert`. In the following example, the prefix is `/root/dsmChain`:
   ```
   /root/dsmChain0.cert
   /root/dsmChain1.cert
   ```

4. Optionally, print the contents of the retrieved server certificate files and verify that the information matches the information in the new server certificate on the DSM server. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later. Issue the following commands:

```
|          mmgskkm print --cert <dsmChain>0.cert
|          mmgskkm print --cert <dsmChain>1.cert
```

| where *dsmChain* is the path and file name prefix of the certificate files that you retrieved in Step 3.
| 5. Issue the following command to add the new DSM server certificate chain to the client keystore. The
| **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later.

```
|    mmgskkm trust --prefix <dsmChain> --out <rkmKeystore> --pwd-file <rkmPassfile>
|    --label <serverLabel>
```

| where:

| **--prefix** *<dsmChain>*
| Is the path and file name prefix of the certificate chain files that you retrieved in Step 3, such
| as /root/dsmChain.

| **--out** *<rkmKeystore>*
| Is the path and file name of the client keystore from Step 1.

| **--pwd-file** *<rkmPassfile>*
| Is the path and file name prefix of the keystore password file that you created in Step 2.

| **--label** *<serverLabel>*
| Is the label under which to store the server certificate in the client keystore.

| **Note:** The label must be unique in the keystore. Also, it cannot be the label of the expired
| server certificate from the DSM key server.
| 6. Copy the updated client keystore to all nodes in the IBM Spectrum Scale cluster.
| 7. Reload the new client keystore by one of the following methods:
| • On any administration node in the cluster, run the **mmchpolicy** command to refresh the current
| policy rules. You do not need to repeat this action on other nodes in the cluster.
| • On each node of the cluster, unmount and mount the file system
| • In IBM Spectrum Scale v4.2.1 and later, issue the following command on each node of the cluster:

```
|      /usr/lpp/mmfs/bin/tsloadikm run
```

| The IBM Spectrum Scale client now trusts the new self-signed Vormetric DSM server certificate.

| ## Renewing expired client certificates
| Follow these instructions to create and renew expired client certificates for the simplified setup, the
| regular setup, and other scenarios.

| See the following topics for detailed instructions:

| **Simplified setup: Creating and registering a new key client:**
| Follow these instructions if you are using SKLM and the simplified setup method. To renew an expired
| client certificate, you must create and register a new key client and deregister the old key client. These
| instructions assume that you want to create a key client c1Client1, deregister the old client c1Client0,
| and register the new key client with tenant devG1 on key server keyserver01.
| 1. Issue the following command to create the key client. Enter a password and a pass phrase when
| prompted:

```
# mmkeyserv client create c1Client1 --server keyserver01
Enter password for the key server keyserver01:
Create a pass phrase for keystore:
Confirm your pass phrase:
```

2. Issue the following command to display information about the current clients. The command output shows that the existing client c1Client0, which has the expired certificate, is registered with tenant devG1 on key server keyserver01. The new client c1Client1 is not registered with a tenant:

```
# mmkeyserv client show
c1Client0
        Label:              c1Client0
        Key Server:         keyserver01
        Tenants:            devG1

c1Client1
        Label:              c1Client1
        Key Server:         keyserver01
        Tenants:            (none)
```

3. Optionally, issue the following command and make a note of the RKM ID that is associated with the old key client. If you reuse the RKM ID of the old key client when you register the new key client, then you do not have to update any of your encryption policy rules that specify the RKM ID:

```
# mmkeyserv tenant show
devG1
        Key Server:         keyserver01.gpfs.net
        Registered Client:  c1Client0
        RKM ID:             keyserver01_devG1
```

See Step 5.

4. Issue the following command to deregister the current key client from the tenant. Notice that this command also deletes the expired certificate:

```
# mmkeyserv client deregister c1Client0 --tenant devG1
Enter password for the key server:
Enter password for the key server of client c1Client0:
mmkeyserv: Deleting the following KMIP certificate with label:
15826749741870337947_devG1_1498047851
```

Note: If you deregister a key client whose certificate is not yet expired, you cannot fetch keys until you register a new key client:

5. Issue the following command to register the new key client with tenant devG1 in key server keyserver01. In the **--rkm-id** parameter, specify a valid RKM ID for the new connection.

Note: Here you can specify the RKM ID of the old key client to avoid having to update encryption policy rules that reference that RKM ID. See Step 3.

```
# mmkeyserv client register c1Client1 --tenant devG1 --rkm-id keyserver01_devG1
Enter password for the key server:
mmkeyserv: [I] Client currently does not have access to the key.
Continue the registration process ...
mmkeyserv: Successfully accepted client certificate
```

For more information, see the topic *mmkeyserv command* in the *IBM Spectrum Scale: Command and Programming Reference*.

The new key client is registered.

All other scenarios: Creating and installing new client credentials:
Follow these instructions if you are using SKLM with a setup method other than the simplified setup.

1. Create a keystore password file that is accessible only by the root user, such as /root/keystore.pwd, and store a password in it. You will need this password file in Step 3.

2. Issue the following command to create new client credentials. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later.

```
mmgskkm gen --prefix <prefix> --cname <cname> --fips <fips>
--nist <nist> --days <validdays> --keylen <keylen>
```

where:

--prefix *<prefix>*
    Is the path and file name prefix of the new certificate files and keystore file.

--cname *<cname>*
    Is the name of the new IBM Spectrum Scale key client. The name can be up to 54 characters in length and can contain alphanumeric characters, hyphen (-), and period (.). In Vormetric DSM, names are not case-sensitive, so it is a good practice not to include uppercase letters.

--fips *<fips>*
    Is the current value of the **FIPS1402mode** configuration variable in IBM Spectrum Scale. Valid values are *yes* and *no*. Issue the following command to see the current value:
```
mmlsconfig FIPS1402mode
```

--nist *<nist>*
    Is the current value of the **nistCompliance** configuration variable in IBM Spectrum Scale. Valid values are **SP800-131A** and **off**. To see the current value, issue the following command:
```
mmlsconfig nistCompliance
```

--days *<validdays>*
    Is the number of days that you want the client certificate to be valid.

--keylen *<keylen>*
    Is the length in bits that you want for the RSA key that is generated.

3. Issue the following command to create a PKCS#12 keystore and to store the client certificate and private key into it. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later.

```
mmgskkm store --cert <certFile> --priv <privFile> --label <label>
--pwd-file <pwd-file> --out <keystore>
```

where:

--cert <certFile>
    Is the client certificate file that you created in Step 2. The name of the file has the format *<prefix>*.cert, where *<prefix>* is the path and file name prefix that you specified in Step 2.

--priv <privFile>
    Is the private key file that you generated in Step 2. The name of the file has the format *<prefix>*.priv, where *<prefix>* is the path and file name prefix that you specified in Step 2.

--label *<label>*
    Is the label under which the new client certificate is stored in the keystore.

--pwd-file *<pwd-file>*
    Is the path and file name of the keystore password file that you created in Step 1.

--out *<keystore>*
    Is the path and file name of the new PKCS#12 keystore. It is a good practice to generate the keystore in the directory /var/mmfs/etc/RKMcerts.

4. Issue the **mmsklmconfig** command to retrieve the RKM server certificate chain. This command is available in IBM Spectrum Scale v4.2.1 and later.

```
mmsklmconfig restcert --host <rkmHost> --port <rkmPort> --prefix <serverPrefix> --keystore <keystore>
--keypass <pwd-file> --fips <fips> --nist <nist>
```

where:

--host *<rkmHost>*
    Is the IP address or host name of the RKM server.

--port *<rkmPort>*
    Is the port of the RKM server:
- For SKLM, the port is the KMIP port, which has a default value of 5696.
- For DSM, the port is the web GUI port, which has a default value of 8445.

--prefix *<serverPrefix>*
    Is the path and file name prefix for the RKM certificate chain.

--keystore *<keystore>*
    Is the path and file name of the PKCS#12 keystore that you created in Step 3.

--keypass *<pwd-file>*
    Is the path and file name of the keystore password file that you created in Step 1.

--fips *<fips>*
    Is the current value of the **FIPS1402mode** configuration variable in IBM Spectrum Scale. Valid values are *yes* and *no*. Issue the following command to see the current value:
```
mmlsconfig FIPS1402mode
```

--nist *<nist>*
    Is the current value of the **nistCompliance** configuration variable in IBM Spectrum Scale. Valid values are **SP800-131A** and **off**. To see the current value, issue the following command:
```
mmlsconfig nistCompliance
```

5. Optionally, print the contents of the server certificate file and verify that the information matches the information that is displayed for the current server certificate in the RKM GUI. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later. You might need to print more than one server certificate file:
```
mmgskkm print --cert <serverPrefix>0.cert
mmgskkm print --cert <serverPrefix>1.cert
```

    where *serverPrefix* is the path and file name prefix of the certificate chain that you specified in Step 4.

6. Issue the following command to add the retrieved certificate to the client keystore. The **mmgskkm** command is available in IBM Spectrum Scale v4.2.1 and later.
```
mmgskkm trust --prefix <serverPrefix> --out <keystore> --pwd-file <pwd-file>
--label <serverLabel>
```

    where:

--prefix *<serverPrefix>*
    Is the path and file name prefix for the RKM certificate chain that you retrieved in Step 4.

--out*<keystore>*
    Is the path and file name of the client keystore that you created in Step 3.

--pwd-file*<pwd-file>*
    Is the path and file name of the keystore password file that you created in Step 1.

--label*<serverLabel>*
    Is the label under which you want to store the RKM certificate chain in the client keystore.

7. Update the RKM stanza for the new client credentials in the /var/mmfs/etc/RKM.conf file. Make sure that the following values are correct:
- The keyStore term specifies the path and file name of the client keystore that you created in Step 3.
- The passphrase term specifies the keystore password from Step 1.
- The clientCertLabel term specifies the label of the new client certificate from Step 3.

8. Copy the updated `/var/mmfs/etc/RKM.conf` file and the new client keystore file to all the nodes of the cluster.
9. Reload the new client keystore by one of the following methods:
   - On any administration node in the cluster, run the **mmchpolicy** command to refresh the current policy rules. You do not need to repeat this action on other nodes in the cluster.
   - On each node of the cluster, unmount and mount the file system.
   - In IBM Spectrum Scale v4.2.1 and later, issue the following command:

     `/usr/lpp/mmfs/bin/tsloadikm run`

     Repeat this action on all the nodes of the cluster.
10. Issue the following command to purge all master encryption keys from the cache of the GPFS daemon:

    `tsctl encKeyCachePurge all`

    This action ensures that subsequent reads and writes to files use the new client credentials.

The new client certificate is installed.

Regular setup: Trusting a new client certificate:
If you have not created and installed new client credentials, follow the instructions in the preceding subsection "All other scenarios: Creating and installing new client credentials" on page 631.

Follow these instructions if you are using SKLM and the Regular setup method and you have created and installed new client credentials.
1. Add the new client certificate to the SKLM list of pending certificates:
   a. On the node that you are configuring for encryption, try to create an encrypted file by doing some action that triggers an encryption policy rule.
   b. The attempt fails because SKLM does not yet trust the new client certificate. However, the attempt causes SKLM to add the new client certificate to the list of pending certificates in the SKLM key server.
2. Verify the RKM ID in the error message from Step 2:
   a. Find the RKM ID that is specified in the error message. In the following example, the RKM ID is keyserver01:

   ```
   # touch /gpfs0/test
   touch: cannot touch `/gpfs0/test': Permission denied
   # tail -n 2 /var/adm/ras/mmfs.log.latest
   Thu Mar 20 14:00:55.029 2014: [E] Unable to open encrypted file: inode 46088, Fileset fs1,
   File System gpfs0.
   Thu Mar 20 14:00:55.030 2014: [E] Error: key 'KEY-326a1906-be46-4983-a63e-29f005fb3a15:keyserver01'
   could not be fetched (RKM reported error -1004).
   ```

   b. Find the RKM ID of the RKM stanza that specifies the new client keystore. The RKM stanza is in the `/var/mmfs/etc/RKM.conf` file.
   c. Verify that the RKM ID from the error message matches the RKM ID of the stanza.
3. Verify the pending client certificate in SKLM:
   a. On the main page of the SKLM graphical user interface, click **Pending client device communication certificates.**.
   b. In the list of certificates, select the new client certificate and click **View**.
   c. Verify that the certificate matches the new client certificate.
   d. If the certificates match, click **Accept and Trust**.
4. Enter a name for the new certificate and click **Accept and Trust** again.
5. Verify that the server accepts the new client certificate:

| SKLM trusts the new client certificate.

| **Regular setup and Vormetric DSM: Trusting a new client certificate:**
| If you have not created a new client certificate, follow the instructions in the preceding subsection "All other scenarios: Creating and installing new client credentials" on page 631.

| Follow these instructions if that you are using a Vormetric Data Security Manager (DSM) key server and the Regular setup method and you have created a new client certificate and imported its information into the current IBM Spectrum Scale policy rules.
| 1. In the DSM web GUI, import the new client certificate into the DSM server. Provide the path and file name of the certificate file that you created in Step 2 and referenced in Step 3 of the subtopic "All other scenarios: Creating and installing new client credentials" on page 631. The path and file name have the format *<prefix>*.cert, where *<prefix>* is the path and file name prefix that you specified in Step 2.
| 2. On the node that you are configuring for encryption, try to create an encrypted file by doing some action that triggers an encryption policy rule. The file is successfully created.

| Vormetric DSM trusts the new client certificate.

Secure deletion

Secure deletion refers to both erasing files from the file system and erasing the MEKs that wrapped the FEKs that were used to encrypt the files.

Securely deleting files in a fileset

After files have been removed from a fileset using standard file system operations (such as **unlink** and **rm**), the tenant administrator might decide to securely delete them. For example, suppose that until that point, the FEKs of all files in the fileset were encrypted with the MEK with key name KEY-old:isklmsrv. To cause the secure deletion of all removed files, the administrator must perform the following steps:

1. Create a new MEK and note its key name (in this example, KEY-new:isklmsrv).
2. Modify the appropriate encryption policy **KEYS** statement in the encryption policy to encrypt new files with the new MEK (for example, KEY-new:isklmsrv) instead of the old one (KEY-old:isklmsrv).
3. Create and apply a migration (rewrapping) policy (**CHANGE ENCRYPTION KEYS**) to scan all files, unwrap the wrapped FEK entries of files that have been wrapped with the old key (KEY-old:isklmsrv), and rewrap them with the new key (KEY-new:isklmsrv); this step ensures that the FEKs of existing files will be accessible in the future.
4. Remove the old key, KEY-old:isklmsrv. This step commits the secure deletion of all files that were previously unlinked (and whose FEKs had therefore not been rewrapped with the new MEK, KEY-new:isklmsrv).
5. On each node that has ever done I/O to a file encrypted with the old key (KEY-old:isklmsrv), run the following command:
   ```
   /usr/lpp/mmfs/bin/tsctl encKeyCachePurge 'KEY-old:isklmsrv'
   ```

From this point on, the new key will be used for encryption, which will be performed transparently to the application.

Note: The **mmdelfs** command will *not* perform any secure deletion of the files in the file system to be deleted. **mmdelfs** only removes all the structures for the specified file system. To securely delete files, you need to perform the following steps:

1. Identify all MEKs currently used to wrap the FEKs of files in the file system to be deleted. If this information is not available through other means, obtain it by doing the following:
   a. Invoke **mmlsattr -n gpfs.Encryption** on all files of the file system.
   b. Parse the resulting output to extract all the distinct key names of the MEKs that are used.

   Note: These are the possible ways that an MEK might be in use in a file system:
   a. The MEK is, or was at some point, specified in an encryption rule in the policy set on the file system.
   b. An FEK rewrap has been run, rewrapping an FEK with another MEK.
2. Determine whether the identified MEKs were used to wrap FEKs in other file systems.

   WARNING: If the same MEKs were used to wrap FEKs in other file systems, deleting those MEKs will result in irreparable data loss in the other file systems where those MEKs are used. Before deleting such MEKs from the key servers, you must create one or more new MEKs and rewrap the files in the other file systems.
3. After appropriately handling any MEKs that were used to wrap FEKs in other file systems (as explained in the warning), delete the identified MEKs from their RKMs.

Secure deletion and encryption key cache purging

The key servers that store the MEKs know how to manage and securely delete keys. After an MEK is gone, all files whose FEKs were encrypted with that MEK are no longer accessible. Even if the data blocks corresponding to the deleted files are retrieved, the contents of the file can no longer be reconstructed, since the data cannot be decrypted.

However, if the MEKs have been cached for performance reasons (so that they do not have to be fetched from the server each time a file is created or accessed), the MEKs must also be purged from the cache to complete the secure deletion.

You can use the following command to purge a given key from the key cache, or to clean the entire cache, of an individual node:

```
/usr/lpp/mmfs/bin/tsctl encKeyCachePurge {Key | all}
```

where:

Key
    is the key ID, specified with the *KeyId*:*RkmId* syntax.

all
    specifies that the entire key cache is to be cleaned.

The scope of this command is limited to the local node and must be run on all nodes that have accessed the MEKs you are purging in order to ensure secure deletion.

Encryption and standards compliance

IBM Spectrum Scale encryption enables the use of FIPS 140-2-certified cryptography and also complies with the recommendations of NIST SP800-131A.

Encryption and FIPS-140-2 certification

Encryption includes an option to enable or disable the use of FIPS 140-2 certified cryptography.

The **FIPS1402mode** configuration variable controls whether the use of crypto-based security mechanisms (if they are to be used at all, per the IBM Spectrum Scale administrator) is to be provided by software modules that are certified according to the requirements and standards described by the Federal

Information Processing Standards (FIPS) 140 Publication Series. When in FIPS 140-2 mode, IBM Spectrum Scale uses the FIPS 140-2 approved cryptographic provider IBM Crypto for C (ICC) (certificate 2420) for cryptography. The certificate is listed on the NIST website.

The value of **FIPS1402mode** can be changed with the **mmchconfig** command. The default value for this variable is **no**. With **FIPS1402mode=no**, Linux nodes will use kernel encryption modules for direct I/O. If a cluster is configured with **FIPS1402mode=yes**, Linux nodes whose kernels are not running in FIPS mode will see a performance degradation when using direct I/O. The GPFS daemon on the node must be restarted in order for the new setting to take place.

Note: In IBM Spectrum Scale V4.2.0 and earlier, in a Power 8, little-endian environment, the setting **FIPS1402mode**=no is required for the following operations:

- File encryption
- Secure communications between nodes. For more information, see the following descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:
    - **-l** *CipherList* parameter of the **mmauth** command
    - **cipherList** parameter of the **mmchconfig** command
- CCR enablement. For more information, see the following descriptions in the *IBM Spectrum Scale: Command and Programming Reference*:
    - **--ccr-enable** parameter of the **mmchcluster** command
    - **--ccr-enable** parameter of the **mmcrcluster** command.

Encryption and NIST SP800-131A compliance

Encryption uses NIST-compliant mechanisms.

The mechanisms that are used by file encryption, including ciphers and key lengths, are compliant with the NIST SP800-131A recommendations. See *NIST Special Publication 800-131A, Revision 1* at http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf.

Encryption in a multicluster environment

If an encrypted file system is made available via remote mounts, then the remote cluster also requires network reachability to the key server. In some deployments, the key servers might be located at the home cluster, but in others one might choose to locate key servers in a high-availability configuration on both home and remote clusters. The order of the servers can be specified in the set of RKM back ends such that the server closest to the local node is accessed first.

Encryption in a Disaster Recovery environment

While setting up multiple key servers in a high-availability configuration is important to ensure that the MEKs remain available, it is especially important in a Disaster Recovery environment. It is a good practice to place at least one key server on each site to ensure that keys remain available if access to an entire site is lost. For more information, see "Identifying multiple RKM back ends in a high-availability configuration" on page 566.

Encryption and backup/restore

GPFS will deliver all data to **mmbackup** and other external backup solutions in **cleartext** whether or not the data is encrypted in GPFS. Any backups that are taken **will not** preserve the encryption status or the encrypted content of the data. Files that are recreated upon restore will be considered for encryption status based on the policy in place on the file system at the time of the restore operation.

Encryption and snapshots

GPFS preserves the encryption status of files when they are copied into global or fileset snapshots. Global snapshot restore will restore files precisely as they are in the snapshot, including FEKs and MEKs. For details about how fileset snapshot restore functions, see the description of the **mmrestorefs** command in the *IBM Spectrum Scale: Command and Programming Reference*.

Encryption and a local read-only cache (LROC) device

IBM Spectrum Scale holds encrypted file data in memory as cleartext. To support this design, IBM Spectrum Scale decrypts encrypted file data as it is read into memory and encrypts file data as it is written into an encrypted file.

By default, IBM Spectrum Scale does not allow cleartext from encrypted files to be copied into an LROC device. The reason is that a security exposure arises when cleartext from an encrypted file is copied into an LROC device. Because LROC device storage is non-volatile, an attacker can capture the cleartext by removing the LROC device from the system and reading the cleartext at some other location.

To enable cleartext from an encrypted file to be copied into an LROC device, you can issue the **mmchconfig** command with the attribute **LROCEnableStoringClearText=yes**. You might choose this option if you have configured your system in some way to remove the security exposure. One such method is to install an LROC device that internally encrypts data that is written into it and decrypts data that is read from it. But see the following warning.

Warning: If you allow cleartext from an encrypted file to be copied into an LROC device, you must take steps to protect the cleartext while it is in LROC storage. One method is to install an LROC storage device that internally encrypts data that is written into it and decrypts data that is read from it. However, be aware that a device of this type voids the IBM Spectrum Scale secure deletion guarantee, because IBM Spectrum Scale does not manage the encryption key for the device.

For more information, see the following links:

Chapter 41, "Local read-only cache," on page 681

The topic *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Encryption requirements and limitations

Learn the requirements and limitations for using encryption.

For encryption requirements, see the topic "Preparation for encryption" on page 563.

Encryption has the following limitations:
- For a multicluster environment, see the topic "Encryption in a multicluster environment" on page 637.
- For a Disaster Recovery environment, see the topic "Encryption in a Disaster Recovery environment" on page 637.
- For backup and restore, see the topic "Encryption and backup/restore" on page 637.
- For snapshots, see the topic "Encryption and snapshots."
- Data for encrypted files is not stored in the inode. For information about data-in-inode, see the topic *Use of disk storage and file structure within a GPFS file system* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- Data from encrypted files is not stored in the highly available write cache (HAWC). For more information, see Chapter 40, "Highly available write cache (HAWC)," on page 677.

| • To avoid a security exposure, by default IBM Spectrum Scale does not allow file data from encrypted
| files, which is held in memory as cleartext, to be copied into an LROC. As a result, a file system in
| which most of the files are encrypted does not take advantage of the performance benefits provided by
| an LROC. However, you can set IBM Spectrum Scale to enable cleartext from encrypted files to be
| copied into an LROC. You might choose this option if you can configure your system to remove the
| security problem.

| **Warning:** If you allow cleartext from an encrypted file to be copied into an LROC, you must take steps
| to protect the cleartext while it is in LROC storage.

| For more information, see the following links:

| "Encryption and a local read-only cache (LROC) device" on page 638

| Chapter 41, "Local read-only cache," on page 681

| The topic *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Chapter 36. Managing certificates to secure communications between GUI web server and web browsers

The IBM Spectrum Scale system supports self-signed and trusted certificates that are provided by a certificate authority (CA) to secure communications between the system and web browser.

During system setup, an initial self-signed certificate is created to use for secure connections between the GUI web servers and web browsers. Based on the security requirements for your system, you can create either a new self-signed certificate or install a signed certificate that is created by the certifying authority. Self-signed certificates can generate web browser security warnings and might not comply with organizational security guidelines. The self-signed certificates are stored in the Liberty profile SSL keystore, which is located at the `resources/security` directory in the server. You can find this directory at the following path: `/opt/ibm/wlp/usr/servers/gpfsgui/resources/security`.

The trusted certificates are created by a third-party certificate authority. These certificate authorities ensure that certificates have the required security level for an organization based on purchase agreements. Trusted certificates usually have higher security controls for encryption of data and do not cause browser security warnings. Trusted certificates are also stored in the Liberty profile SSL keystore.

Major web browsers trust the CA-certified certificates by default and therefore they can confirm that the certificate received by the GUI server can be trusted. You can either buy a signed certificate from a trusted third-party authority or create your own certificate and get it certified. You can use both self-signed and trusted certificates. However, using a trusted is the preferred way because the browser trusts this certificate automatically without any manual interventions.

Obtaining and importing a signed-certificate from a trusted certificate authority

You need to perform the following steps to obtain and import a signed-certificate from a trusted certificate authority:

1. Generate a private key by issuing the following command:

   ```
   openssl genrsa -out <nameOfYourKey>.key 2048
   ```

2. Generate the certificate request as shown in the following example:

   ```
   openssl req -new -key <nameOfYourKey>.key -out <nameOfYourKey>.csr
   ```

   The system prompts you to enter the following details:

   ```
   Country Name (2 letter code) [XX]:
   State or Province Name (full name) []:
   Locality Name (eg, city) [Default City]:
   Organization Name (eg, company) [Default Company Ltd]:
   Organizational Unit Name (eg, section) []:
   Common Name (eg, your name or your server's hostname) []:
   Email Address []:
   Please enter the following 'extra' attributes to be sent with your certificate request
   A challenge password []:
   An optional company name []:
   ```

3. Send the certificate request to a trusted certificate authority to get a certificate file.

4. Create a PKCS12 store containing the certificate as shown in the following example:

   ```
   openssl pkcs12 -export -in <yourCertificateFile> -inkey <nameOfYourKey>.key
   <nameOfYourPKCS12File>.p12
   ```

   The system prompts to set the export password as shown in the following example:

   ```
   Enter export Password: <yourPassword>
   Verifying - Enter export Password: <yourPassword>
   ```

5. Generate a Java™ keystore file (.jks) by using the keytool. It is stored in the following directory: `/usr/lpp/mmfs/java/jre/bin`. Ensure that you set the paths to the Java keystore file properly in the system. Issue the following commands to generate a Java keystore file.

```
<PathToKeytool>/keytool -importkeystore -srckeystore
<NameOfYourPKCS12File>.p12 -destkeystore
<NameOfYourJKSFile>.jks -srcstoretype pkcs12
```

The system prompts you to enter the destination keystore password. You need to use the same password that you used while creating the PKCS12 store.

```
Enter destination keystore password: <yourPassword>
Re-enter new password: <yourPassword>
Enter source keystore password: <yourPassword>
```

6. Copy the new Java keystore file to the directory named `security`, which is stored in the GUI server. This directory is located at the following location in the GUI server: `/opt/ibm/wlp/usr/servers/gpfsgui/resources/security`. It is the default place where keystore files are stored.

```
cp <NameOfYourJKSFile>.jks <pathToSecurityDir>
```

7. You need to define the password to access the Java Keystore file in the `server.xml` file, which is stored in the GUI server. If you want to encode your password so that it does not get stored in plain text, use a security utility, which is stored in the following directory: `/opt/ibm/wlp/bin`. The supported encodings are XOR and AES.

```
<PathToSecurityUtility>/securityUtility encode --encoding=<xor or aes> <yourPassword>
```

8. Edit keystore entry of the `server.xml` file for the GUI server to enable the new key. The `server.xml` file is located at the following location: `/opt/ibm/wlp/usr/servers/gpfsgui`. The following entry is on one line:

```
<keyStore id="defaultKeyStore" password="<yourPassword> or <yourEncodedPassword>"
 location="<nameOfYourJKSFile>.jks" />
```

9. Restart the IBM Spectrum Scale management GUI by issuing the following commands:

For RHEL7.x, SLES12.x, or Ubuntu 16.x: `systemctl restart gpfsgui`
For RHEL6.x: `service gpfsgui restart`

Chapter 37. Securing protocol data

The data cannot be secured only by authenticating and authorizing the users to access the data. You also need to ensure that the communication channel that is used to raise authentication requests and data transfer is secured. The security features associated with the protocols that you use to store and access data also help to provide data in transit security for the protocol data.

The secured data access by clients through protocols is achieved through the following two steps:

1. Establishing secured connection between the IBM Spectrum Scale system and the authentication server.

   When the client raises an authentication request to access the data, the IBM Spectrum Scale system interacts with the external authentication servers like Active Directory or LDAP based on the authentication configuration. You can configure the security services like TLS and Kerberos with the external authentication server to secure the communication channel between the IBM Spectrum Scale system and the external authentication server.

2. Securing the data transfer.

   The actual data access wherein the data transfer is made secured with the security features that are available with the protocol that you use to access the data.

The following diagram depicts the data in transit security implementation in the IBM Spectrum Scale system.

Figure 22. Implementation of data in transit security for protocol data

Secured connection between the IBM Spectrum Scale system and the authentication server

You can configure the following authentication servers to configure file and object access:

- Microsoft Active Directory (AD)
- Lightweight Directory Access Protocol (LDAP)
- Keystone

AD and LDAP can be used as the authentication server for both file and object access. Configuring the Keystone server is a mandatory requirement for the object access to function. The keystone needs to interact with the authentication server to resolve the authentication requests. You can configure either an internal or external keystone server for object access. The following table lists the security features that are used to secure the corresponding authentication server.

Table 56. Security features that are used to secure authentication server.

Authentication server	Supported protocols	Security features
Active Directory	File and Object	Kerberos for file and TLS for object.

Table 56. Security features that are used to secure authentication server (continued).

Authentication server	Supported protocols	Security features
LDAP	File and Object	Both TLS and Kerberos for file and only TLS for object.
Keystone	Object	SSL certificate to enable HTTPS connection

Secured data transfer

The secured data transfer over the network is based on the security features available with the protocols that are used to access the data.

Secured SMB data transfer

SMB protocol version 3 and later has the following capabilities to provide tighter security for the data transfers:
1. Secured dialect negotiation
2. Improved signing
3. Secured transmission

The dialect negotiation is used to identify the highest level dialect both server and client can support. The system administrator can enable SMB encryption by using the `smb encrypt` setting at the export level. The following three modes are available for the secured SMB access:
* Automatic
* Mandatory
* Disabled

When the SMB services are enabled, the SMB encryption is enabled in the automatic mode by default.

Note: SMB supports per-export encryption, which allows the administrators to selectively enable or disable encryption per SMB share.

Secured NFS data transfer

The following security methods are used with NFSV4 protocol:
1. Enabling squashing

   Any file requests that are made by the root user on the client system is considered as a potential threat. By default, root user requests are treated as if it is made by the user on the server. If you disable squashing, the root user on the client gets the same level of access to files on the system as the root user on the server. You can disable squashing if, for example, you want to run an administrative task on the client system that has the exported directories that are stored on it.
2. Using Kerberos

   Kerberos is a network authentication protocol that ensures secure communication over a network. You can use Kerberos instead of local UNIX UIDs and GIDs to authenticate users. Kerberos can operate in the following modes to provide improved security:
   * **Kerberos v5:** Authentication only
   * **Kerberos v5 with integrity:** Authentication and data integrity
   * **Kerberos v5 with privacy:** Authentication and encryption of data traffic between the client and the server. Most secure, but it might cause some performance issues because of the heavy processing required for encryption.
3. Enabling port security

You can enable or disable the port security in all communications between the client and the server. When port security is enabled, the system does not allow access to the requests that originate from ports where the port number is greater than the hardcoded threshold value of 1024. NFS port security is mapped to the "PRIVILEGEDPORT" configuration parameter. This option is available in the `mmnfs config` command (for changing the system default) and in the `mmnfs export` command (for changing the individual exports). "PRIVILEGEDPORT" by default is set to "False", which means that there is no limit on the accepted source ports or that the port security is not enabled. To enable port security, enable the "PRIVILEGEDPORT" parameter with `mmnfs config change` or `mmnfs export create/change` command.

Secured object data access

The IBM Spectrum Scale system provides access to the Object Storage with the help of OpenStack Keystone Identity Service. The Keystone server that is provided by IBM Spectrum Scale is recommended to be used only for IBM Spectrum Scale Object workload.

For secure communication between the clients and the IBM Spectrum Scale Object, the system administrator needs to configure HAProxy for SSL termination, traffic encryption, and load balancing of the requests to IBM Spectrum Scale Object. The HAProxy needs to be set up on an external system that is not a part of the IBM Spectrum Scale cluster. For more information on how to configure HAProxy, see the documentation of the corresponding Linux distribution that you selected.

Planning for protocol data security

It is recommended to adhere to the following best practices when you plan to set up data security:
- When Windows clients use SMB 3.0, always configure SMB share with `smb encrypt = mandatory`.
- Avoid clients who use SMB 2.1 from accessing SMB data.
- Always enforce to use UNC with NetBIOS name of the cluster to access SMB data.
- To ensure NFSV4 encryption, use an authentication method that is configured with Kerberos.

Configuring protocol data security

The data security features associated with protocols facilitate to configure a secured way for the clients to raise the data access request and to transfer data from the IBM Spectrum Scale system to the client system.

Enabling secured connection between the IBM Spectrum Scale system and authentication server

You need to secure the communication channel between the IBM Spectrum Scale system and authentication server to secure the authentication server and hence to prevent unauthorized access to data and other system resources.

Securing AD server

To secure the AD server that is used for file access, configure it with Kerberos and to secure AD used for object access, configure it with TLS.

In the AD-based authentication for file access, Kerberos is configured by default. The following steps provide an example on how to configure TLS with AD, while it is used for object access.

1. Ensure that the CA certificate for AD server is placed under /var/mmfs/tmp directory with the name *object_ldap_cacert.pem*, specifically on the protocol node where the command is run. Perform validation of CA cert availability with desired name at required location as shown in the following example:

```
# stat /var/mmfs/tmp/object_ldap_cacert.pem
File: /var/mmfs/tmp/object_ldap_cacert.pem
Size: 2130 Blocks: 8 IO Block: 4096 regular file
```

```
Device: fd00h/64768d Inode: 103169903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 12:37:34.088837381 +0530
Modify: 2015-01-23 12:16:24.438837381 +0530
Change: 2015-01-23 12:16:24.438837381 +0530
```

2. To configure AD with TLS authentication for object access, issue the **mmuserauth service create**
   command:

```
# mmuserauth service create --type ad --data-access-method object
--user-name "cn=Administrator,cn=Users,dc=IBM,dc=local"  --base-dn "dc=IBM,DC=local"
--enable-server-tls --ks-dns-name myKeystoneDnsName
--ks-admin-user admin --servers myADserver
--user-id-attrib cn --user-name-attrib sAMAccountName
--user-objectclass organizationalPerson --user-dn "cn=Users,dc=IBM,dc=local"
--ks-swift-user swift
Object configuration with LDAP (Active Directory) as identity
backend is completed successfully.
Object Authentication configuration completed successfully.
```

   Note: The value that you specify for **--servers** must match the value in the TLS certificate. Otherwise
   the command fails.

3. To verify the authentication configuration, use the **mmuserauth service list** command as shown in
   the following example:

```
# mmuserauth service list
FILE access not configured
PARAMETERS              VALUES
-------------------------------------------------

OBJECT access configuration: AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       true
ENABLE_KS_SSL           false
USER_NAME               cn=Administrator,cn=Users,dc=IBM,dc=local
SERVERS                 myADserver
BASE_DN                 dc=IBM,DC=local
USER_DN                 cn=users,dc=ibm,dc=local
USER_OBJECTCLASS        organizationalPerson
USER_NAME_ATTRIB        sAMAccountName
USER_ID_ATTRIB          cn
USER_MAIL_ATTRIB        mail
USER_FILTER             none
ENABLE_KS_CASIGNING     false
KS_ADMIN_USER           admin
```

Securing LDAP server

To secure the LDAP server that is used for file access, configure it with TLS and Kerberos and to secure
LDAP server that is used for object access, configure it with TLS.

Provide examples of how to configure LDAP with TLS and Kerberos to secure the LDAP server when it
is used for file and object access.

1. To configure LDAP with TLS and Kerberos as the authentication method for file access, issue the
   mmuserauth service create command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method file
--servers es-pune-host-01 --base-dn dc=example,dc=com
--user-name cn=manager,dc=example,dc=com
--netbios-name ess --enable-server-tls --enable-kerberos
--kerberos-server es-pune-host-01 --kerberos-realm example.com
```

   The system displays the following output:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       true
ENABLE_KERBEROS         true
USER_NAME               cn=manager,dc=example,dc=com
SERVERS                 es-pune-host-01
NETBIOS_NAME            ess
BASE_DN                 dc=example,dc=com
USER_DN                 none
GROUP_DN                none
NETGROUP_DN             none
USER_OBJECTCLASS        posixAccount
GROUP_OBJECTCLASS       posixGroup
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
KERBEROS_SERVER         es-pune-host-01
KERBEROS_REALM          example.com

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

2. To configure LDAP with TLS as the authentication method for object access, issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method object
--user-name "cn=manager,dc=essldapdomain"
--base-dn dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com --enable-server-tls
--ks-dns-name c40bbc2xn3 --ks-admin-user mamdouh --servers 192.0.2.11
--user-dn "ou=People,dc=essldapdomain" --ks-swift-user swift
```

The system displays the following output:

```
Object configuration with LDAP as identity backend is completed successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access not configured
PARAMETERS              VALUES
-------------------------------------------------

OBJECT access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       true
ENABLE_KS_SSL           false
USER_NAME               cn=manager,dc=essldapdomain
SERVERS                 192.0.2.11
BASE_DN                 dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com
USER_DN                 ou=people,dc=essldapdomain
USER_OBJECTCLASS        posixAccount
```

```
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
USER_MAIL_ATTRIB        mail
USER_FILTER             none
ENABLE_KS_CASIGNING     false
KS_ADMIN_USER           mamdouh
```

Securing Keystone server

The Keystone server that is used by the IBM Spectrum Scale system supports SSL. The SSL certificate
provides secure communication while resolving the authentication requests. When Keystone is configured
with authentication servers such as LDAP or AD, the system can be configured to establish a secured
communication between AD or LDAP and Keystone by using TLS encryption. For more information on
configuring AD or LDAP-based authentication with TLS, see the **mmuserauth service create** command.
The IBM Spectrum Scale for Object Storage can also be configured with an external Keystone server. If
the external Keystone server contains SSL certificate in place, then the system administrator can configure
secured communication with the IBM Spectrum Scale system by following some manual steps.

The following is an example on how to configure secured object access.

1. Remove the object authentication and the ID mapping:

   ```
   /usr/lpp/mmfs/bin/mmuserauth service remove  --data-access-method object
   /usr/lpp/mmfs/bin/mmuserauth service remove  --data-access-method object --idmapdelete
   mmuserauth service list
   ```

   The system displays the following output:

   ```
   FILE access not configured
   PARAMETERS              VALUES
   -------------------------------------------------
   OBJECT access not configured
   PARAMETERS              VALUES
   -------------------------------------------------
   ```

2. Copy the CA certificate on the node on which the **mmuserauth** command is being run. The name and
   the path of the CA certificate on the current node is /var/mmfs/tmp/ks_ext_cacert.pem.

3. Configure object authentication by using the **mmuserauth service create** command with the
   --enable-ks-ssl option:

   ```
   mmuserauth service create --data-access-method object --enable-ks-ssl --type
   userdefined --ks-ext-endpoint https://externalkeystoneserver:35357/v3
   --ks-swift-user swift
   ```

4. Run the **mmuserauth service list** command to verify the configuration:

   ```
   mmuserauth service list
   FILE access not configured
   PARAMETERS              VALUES
   -------------------------------------------------
   OBJECT access configuration : USERDEFINED
   PARAMETERS              VALUES
   -------------------------------------------------
   ```

Securing data transfer

The data in transit security is configured by using the security features that are available with the
protocol that is used for data I/O.

Securing NFS data transfer

Securing the NFS data transfer over the network is achieved by using the Kerberos-based encryption that
is available with NFSV4 protocol. You can use Kerberos to encrypt the data that is transferred over the
network and also to secure the communication with the authentication server.

The following example shows how to enable data security to ensure secured NFS data transfer.

1. Create a keytab file for protocol nodes in IBM Spectrum Scale cluster. To create a keytab file, you need to create a principal nfs/<node-fqdn> for each protocol node. Issue the following commands on the system that hosts the KDC server. In the following example, the sample commands are submitted on the Linux system that hosts MIT KDC server:

```
$ addprinc -randkey nfs/<protocol-node1-fqdn>
$ addprinc -randkey nfs/<protocol-node2-fqdn>

.....

&mldr; $ addprinc -randkey nfs/<protocol-nodeN-fqdn>
$ ktadd -k /tmp/krb5.keytab nfs/<protocol-node1-fqdn>
$ ktadd -k /tmp/krb5.keytab nfs/<protocol-node2-fqdn>

.....

&mldr; $ ktadd -k /tmp/krb5.keytab nfs/<protocol-nodeN-fqdn>
```

2. Ensure that the keytab file that is created is placed under the /tmp directory as *krb5.keytab*, specifically on the node where the IBM Spectrum Scale authentication commands are submitted. Perform validation of keytab file availability with the required name and location:

```
# stat /tmp/krb5.keytab
File: /tmp/krb5.keytab
Size: 502 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169898 Links: 1
Access: (0600/-rw-------) Uid: ( 0/ root) Gid: ( 0/ root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 14:31:18.244837381 +0530
Modify: 2015-01-23 12:45:05.475837381 +0530
Change: 2015-01-23 12:45:05.476837381 +0530
Birth: -
```

3. Issue the **mmuserauth service create** command on the IBM Spectrum Scale protocol node as shown in the following example:

```
#  mmuserauth service create --data-access-method file --type ldap
--servers 192.0.2.17 --base-dn dc=example,dc=com
--user-name "cn=manager,dc=example,dc=com"  --enable-kerberos
--kerberos-server 192.0.2.17 --kerberos-realm example.com --netbios-name cktest
File Authentication configuration completed successfully.
```

4. Issue the **mmuserauth service list** command to see the current authentication configuration as shown in the following example:

```
# mmuserauth service list
ILE access configuration : LDAP
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND   false
ENABLE_SERVER_TLS       false
ENABLE_KERBEROS         true
USER_NAME               cn=manager,dc=example,dc=com
SERVERS                 9.118.46.17
NETBIOS_NAME            cktest
BASE_DN                 dc=example,dc=com
USER_DN                 none
GROUP_DN                none
NETGROUP_DN             none
USER_OBJECTCLASS        posixAccount
GROUP_OBJECTCLASS       posixGroup
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
KERBEROS_SERVER         9.118.46.17
KERBEROS_REALM          example.com


OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

5. Create Kerberos exports with krb5, krb5i, and krb5p security features on the IBM Spectrum Scale
   node.

   ```
   # mmcrfileset gpfs0 krb5
   Fileset krb5 created with id 2 root inode 47898.

   # mmlinkfileset gpfs0 krb5 -J /ibm/gpfs0/krb5
   Fileset krb5 linked at /ibm/gpfs0/krb5

   # mmnfs export add /ibm/gpfs0/krb5 --client \
       "*(ACCESS_TYPE=RW,SQUASH=no_root_squash,SECTYPE=krb5)"
   The NFS export was created successfully.

   # mmcrfileset gpfs0 krb5i
   Fileset krb5i created with id 3 root inode 47900.

   # mmlinkfileset gpfs0 krb5i -J /ibm/gpfs0/krb5i
   Fileset krb5i linked at /ibm/gpfs0/krb5i

   # mmnfs export add /ibm/gpfs0/krb5i --client \
       "*(ACCESS_TYPE=RW,SQUASH=no_root_squash,SECTYPE=krb5i)"
   The NFS export was created successfully.

   # mmcrfileset gpfs0 krb5p
   Fileset krb5p created with id 4 root inode 47895.

   # mmlinkfileset gpfs0 krb5p -J /ibm/gpfs0/krb5p
   Fileset krb5p linked at /ibm/gpfs0/krb5p

   # mmnfs export add /ibm/gpfs0/krb5p --client \
       "*(ACCESS_TYPE=RW,SQUASH=no_root_squash,SECTYPE=krb5p)"
   The NFS export was created successfully.

   # mmnfs export list
   ```

   The system displays output similar to this:

   ```
   Path                  Delegations   Clients
   ------------------------------------------
   /ibm/gpfs0/krb5    none           *
   /ibm/gpfs0/krb5i   none           *
   /ibm/gpfs0/krb5p   none           *
   /ibm/gpfs0/nfsexp1 none           *
   ```

6. Issue the **mmnfs export list** command with krb5 option to see the authentication only configuration.

   ```
   # mmnfs export list --nfsdefs /ibm/gpfs0/krb5
   ```

   The system displays output similar to this:

   ```
   Path            Delegations Clients Access_Type Protocols Transports Squash       Anonymous_uid Anonymous_gid SecType PrivilegedPort Export_id DefaultDelegation Manage_Gids NFS_Commit
   -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
   /ibm/gpfs0/krb5 none        *       RW          3,4       TCP        NO_ROOT_SQUASH -2          -2            KRB5    FALSE          2         none              FALSE       FALSE
   ```

7. Issue the **mmnfs export list** command with krb5i option to see the authentication and data integrity
   configuration.

   ```
   # mmnfs export list --nfsdefs /ibm/gpfs0/krb5i
   ```

   The system displays output similar to this:

   ```
   Path             Delegations   Clients Access_Type Protocols Transports Squash       Anonymous_uid Anonymous_gid SecType PrivilegedPort Export_id DefaultDelegation Manage_Gids NFS_Commit
   -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
   /ibm/gpfs0/krb5i none          *       RW          3,4       TCP        NO_ROOT_SQUASH -2         -2            KRB5I   FALSE          3         none              FALSE       FALSE
   ```

8. Issue the **mmnfs export list** command with krb5p option to see the authentication and privacy
   configuration.

   ```
   # mmnfs export list --nfsdefs /ibm/gpfs0/krb5p
   ```

   The system displays output similar to this:

   ```
   Path             Delegations   Clients Access_Type Protocols Transports Squash       Anonymous_uid Anonymous_gid SecType PrivilegedPort Export_id DefaultDelegation Manage_Gids NFS_Commit
   -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
   /ibm/gpfs0/krb5p none          *       RW          3,4       TCP        NO_ROOT_SQUASH -2         -2            KRB5P   FALSE          4         none              FALSE       FALSE
   ```

Securing SMB data transfer

Secured SMB data transfer can be enabled when you are using SMB3 and later.

You can either enable or disable encryption of the data in transit by using the **mmsmb export add** command as shown in the following example:

```
# mmsmb export add secured_export  /ibm/gpfs0/secured_export --option "smb encrypt=mandatory"
```

Secured object data transfer

For secure communication between the clients and the IBM Spectrum Scale Object, the system administrator needs to configure HAProxy for SSL termination, traffic encryption, and load balancing of the requests to IBM Spectrum Scale Object. The HAProxy needs to be set up on an external system that is not a part of the IBM Spectrum Scale cluster. For more information on how to configure HAProxy, see the documentation of the corresponding Linux distribution that you selected.

Data security limitations

The following are the protocol data security limitations:

- The file system encryption feature is not supported for securing file and object protocol data.
- The SMB encryption is available only on SMB3 and later. All the limitations that are identified by Microsoft also apply to SMB encryption. There are no SMB encryption limitations that are specific to IBM Spectrum Scale.
- Delegations cannot be used with NFS in the Kerberos environment, because they cause the NFSV4 server to crash. If you use NFS in the Kerberos environment, you should disable delegations.

Chapter 38. Cloud services: Transparent cloud tiering and Cloud data sharing

This topic provides a brief description about managing your cloud storage using IBM Spectrum Scale.

Administering Transparent cloud tiering and Cloud data sharing services

This topic provides a brief description on how to manage Transparent cloud tiering and Cloud data sharing in the IBM Spectrum Scale cluster.

Stopping Cloud services software

This topic describes the procedure for stopping the Cloud services software.

To stop Cloud services on all Transparent cloud tiering nodes in a cluster, issue the following command:

```
mmcloudgateway service stop -N alltct
```

To stop the Cloud services on a specific node or a list of nodes, issue a command according to this syntax:

```
mmcloudgateway service stop [-N alltct {Node[,Node...] | NodeFile | NodeClass}]
```

For example, to stop the service on the node, 10.11.12.13, issue this command:

```
mmcloudgateway service stop -N 10.11.12.13
```

You can run this command on any node in the cluster.

Note: Before you stop Cloud services, ensure that no migration or recall operation is running on the system where the service is stopped. You can find out the status of the migration or recall operation from the GUI metrics.

Monitoring the health of Cloud services software

Use the **mmcloudgateway** command to monitor the health of cloud services.

To monitor the status of Cloud services, issue a command according to this syntax:

```
mmcloudgateway service status [-N {alltct | Node[,Node...] | NodeFile | NodeClass}]
   [--cloud-storage-access-point-name CloudStorageAccessPointName] [-Y]
```

For example,

- To check the status of all available Transparent cloud tiering nodes in your cluster, issue this command:

  ```
  mmcloudgateway service status -N alltct
  ```

The system displays output similar to this:

```
Cloud Node Class:  tct
=====================

Cloud Service              Status      Reason
---------------------------------------------
swift-service              ENABLED

                           Server      Account /   Container /
Node   Daemon node name    Status      CSAP        File System/Set         Status    Reasons
```

```
-------------------------------------------------------------------------------------------------------------------
    1  vm597.pk.slabs.ibm.com                   STARTED    swift-account  swift-pair
                                                           swift-point  /gpfs/                              ONLINE
    2  vm482.pk.slabs.ibm.com                   STARTED    swift-account  swift-pair
                                                           swift-point  /gpfs/                              ONLINE
```

- To check the status of all available Transparent cloud tiering nodes in a specific node class (TctNode), issue this command:

```
mmcloudgateway service status -N TcTNode
```

The system displays output similar to this:

```
Cloud Node Class:  TcTNode
======================


  Cloud Service             Status      Reason
  ---------------------------------------------
  cs1                       ENABLED
  cs2                       ENABLED


                                       Server      Account /   Container /
  Node  Daemon node name               Status      CSAP        File System/Set              Status    Reasons
  --------------------------------------------------------------------------------------------------------------
     1  pk.slabs.ibm.com               STARTED     swift-next  cpairnext
                                                   csapnext    /gpfs/fs1/                    ONLINE
                                                   swift-new
                                                   csap                                      ONLINE
```

- To check the status of all available Transparent cloud tiering nodes in a specific CSAP, issue this command:

```
mmcloudgateway service status -N TctNode --cloud-storage-access-point-name swift-point
```

The system displays output similar to this:

```
Cloud Node Class: TctNode
======================

Cloud Service Status Reason
--------------------------
swift-service ENABLED

       Server Account / Container /
Node   Daemon node name   Status   CSAP                                      File System/Set   Status   Reasons
--------------------------------------------------------------------------------------------------------------
1      jupiter-vm1192     STARTED  swift-account swift-pairswift-point        /gpfs/            Online   ONLINE
```

For more information on all the available statuses and their description, see the *Transparent Cloud Tiering status description* topic in *IBM Spectrum Scale: Command and Programming Reference.*

GUI navigation

To work with this function in the GUI,

- Log on to the IBM Spectrum Scale GUI and select **Files >Transparent cloud tiering**
- Log on to the IBM Spectrum Scale GUI and select **Monitoring>Statistics**

Additionally, you can check the Cloud services status by using the `mmhealth node show CLOUDGATEWAY` command.

During a rolling upgrade from version 4.2.3 to version 5.0, the `mmhealth` command and GUI display the 4.2.3-style events with suggestions about addressing the output of the `mmcloudgateway` command. If you specify mmcloudgateway `service status` on a version 4.2.3 node, you can see the corresponding referred output. After you upgrade all nodes and the minimum release level to version 5.0 or greater, the `mmhealth` command and GUI consistently show version 5.0-style events (with references to the `mmcloudgateway` command output).

| Health monitoring of 5.0 Transparent cloud tiering is supported on version 4.2.3 client if there is:
| • One cloud account configured
| • One container pairset configured
| • One cloud storage access point (CSAP)

Checking the Cloud services version

This topic describes how to check the Cloud services versions of each node in a node class.

CLI commands do not work on a cluster if all nodes in a node class are not running the same version of the Cloud services. For example, you have three nodes (node1, node2, node3) in a node class (TCTNodeClass1). Assume that the Cloud services version of node1 is 1.1.1, of node2 is 1.1.1, and of node3 is 1.1.2. In this case, the CLI commands specific to 1.1.2 do not work in the `TCTNodeClass1` node class.

| To check the service version of all Transparent cloud tiering nodes in a cluster, issue the following
| command:
| ```
| mmcloudgateway service version -N alltct
| ```

To check for service versions associated with Cloud services nodes, issue a command according to this syntax:
```
mmcloudgateway service version  [-N {Node[,Node...] | NodeFile | NodeClass}]
```

For example, to display the Cloud services version of the nodes, `node1` and `node2`, issue the following command:
```
mmcloudgateway service version -N node1,node2
```

The system displays output similar to this:
```
Node  Cloud node name                 TCT Type              TCT Version
-------------------------------------------------------------------------------
   8  node1                           Client                1.1.1
   9  node2                           Server                1.1.2
```

| To display the Cloud services version of each node in a node class, `TCT`, issue the following command:
| ```
| mmcloudgateway service version -N TCT
| ```

| The system displays output similar to this:
| ```
| Cluster minReleaseLevel: 5.0.0.0
|
| Node Daemon node name   TCT Type   TCT Version   Equivalent Product Version
| ---
| 1   jupiter-vm1192.pok.stglabs.ibm.com Server   1.1.4   5.0.0.0
| ```

| To display the client version of each node, issue the following command on the client node:
| ```
| mmcloudgateway service version
| ```

| The system displays output similar to this:
| ```
| Cluster minReleaseLevel: 5.0.0.0
|
| Node Daemon node name TCT Type TCT Version Equivalent Product Version
| ---
| 4 jupiter-vm649.pok.stglabs.ibm.com Client 1.1.4 5.0.0.0
| ```

| To verify the client version of a particular node, issue the following command:
| ```
| mmcloudgateway service version -N jupiter-vm717
| ```

The system displays output similar to this:

```
Cluster minReleaseLevel: 5.0.0.0

Node Daemon node name TCT Type TCT Version Equivalent Product Version
--------------------------------------------------------------------------------------------
3 jupiter-vm717.pok.stglabs.ibm.com Client 1.1.4 5.0.0.0
```

To check for all nodes in a node class, issue the following command:

```
mmcloudgateway service version -N tct
```

The system displays output similar to this:

```
Cluster minReleaseLevel: 5.0.0.0

Node Daemon node name TCT Type TCT Version Equivalent Product Version
--------------------------------------------------------------------------------------------
2 jupiter-vm482.pok.stglabs.ibm.com Server 1.1.4 5.0.0.0
1 jupiter-vm597.pok.stglabs.ibm.com Server 1.1.4 5.0.0.0
```

Note: If you run this command from a remote system, Cloud services version 4.2.1 clients will be designated as a "Non TCT Node" in the command output for the specific node due to a limitation in the current release. Therefore, you need to manually track and check the version of Cloud services version 4.2.1 clients until these nodes are upgraded to 4.2.2.

Administering files for Transparent cloud tiering

You can administer files on the cloud storage account when you use Transparent cloud tiering.

When you use Transparent cloud tiering, there are maintenance tasks that must be done. It is recommended that you put them in a scheduler to make sure that the maintenance activity is performed. Since these maintenance activities affect overall data throughput, schedule them one at time (do not schedule them simultaneously) during non-peak demand times.

1. Reconcile your files once a month to make sure that the cloud directories that are maintained by Transparent cloud tiering services and the file system are synchronized.
2. Do a full backup of the cloud directory once a month to allow for faster and cleaner handling of disaster recovery and service problems.
3. Run the cloud destroy utility to remove files that are deleted from the file system from the cloud system.

These steps must be run for each Transparent cloud tiering container in the file system.

Note: You do not have to perform these actions for inactive containers that are not being migrated to (and have no delete activity).

See the information below for detailed instructions on how to perform these maintenance steps.

Applying a policy on a Transparent cloud tiering node

This topic provides description with an example about creating an ILM policy for tiering and then applying this policy to a Transparent cloud tiering node.

After a cloud account is configured, you can apply an ILM policy file to configure a cloud storage tier. The policy configuration is done by using IBM Spectrum Scale standard ILM policy query language statements.

For more information on ILM policies, see Chapter 25, "Information lifecycle management for IBM Spectrum Scale," on page 355 .

You must create a policy and then apply this policy on the Cloud services node for the ILM-based migration and recall to work for the cloud storage tier.

Note: Administrators must consider appropriate high and low disk utilization threshold values that are applicable in the data center environment.

A sample policy rule and the steps to apply the policy on a node are as follows:

```
/* Sample policy.rules file for using Gateway functionality */
/* Define an external pool for the off-line storage */
define(
exclude_list,
  (
   FALSE
   OR PATH_NAME LIKE '%/.mcstore/%'
  )
)
define(
access_age,
(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME))
)
define(
mb_allocated,
(INTEGER(KB_ALLOCATED / 1024))
)
define(
weight_expression,
(CASE
 /*=== The file is very young, the ranking is very low ===*/
 WHEN access_age <= 1 THEN 0
 /*=== The file is very small, the ranking is low ===*/
 WHEN mb_allocated < 1 THEN access_age
 /*=== The file is resident and large and old enough,
 the ranking is standard ===*/
 ELSE mb_allocated * access_age
 END)
 )
/* Define an external pool for the off-line storage */
RULE EXTERNAL POOL 'mcstore' EXEC '/opt/ibm/MCStore/bin/mcstore' OPTS '-F'
/* Define migration rule with a threshold to trigger low space events
and move data to the external off-line pool. When on-line usage
exceeds 25% utilization, it will move the coldest files to off-line storage
until the on-line usage is reduced to 20% utilization level. Only files that have
data on-line are eligible for migration. */
RULE 'MoveOffline' MIGRATE FROM POOL 'system'
THRESHOLD(25,20)
WEIGHT(weight_expression)
TO POOL 'mcstore'
WHERE(KB_ALLOCATED > 0) AND NOT(exclude_list)
/* Define default placement rule */
RULE 'Placement' SET POOL 'system'
```

For more information on how to work with the external storage pools and related policies, see "Working with external storage pools" on page 394.

Note: Ensure that only a single instance of the policy is applied to migrate data to the external cloud storage pool. This avoids any potential locking issues that might arise due to multiple policy instances that try to work on the same set of files.

To ensure proper invocation of the policy on reaching threshold limits, see Threshold based migration using callbacks example.

In the sample policy, the 'OpenRead' & 'OpenWrite' rule sections represent the transparent recall of a migrated or non-resident file. Transparent cloud tiering software adds its own extended attributes

(dmapi.MCEA) to each file it processes. Displacement 5 in the extended attributes indicate the resident state of the file. If it is 'N' (non-resident), the policy issues a recall request to bring back the data from the cloud storage to the local file system for the requested Read or Write operation.

To apply a threshold policy to a file system, see "Using thresholds to migrate data between pools" on page 392.

IBM Spectrum Scale also gives administrators a way to define policies to identify the files for migration, and apply those policies immediately using the `mmapplypolicy` command. This is different from the threshold-based policies (which are applied by using the `mmchpolicy` command). The Transparent cloud tiering service currently does not support parallelism in migrating files simultaneously, but parallelism in the `mmapplypolicy` command can be used to improve the overall throughput. Additionally, parallelism can be achieved by using an ILM policy to migrate data or by driving separate, parallel CLI commands.

A sample command to apply a policy is given here:

```
mmapplypolicy gpfs0 -P <rules.file> -m 24 -B 100
```

where,
- *gpfs0* indicates the IBM Spectrum Scale system
- *-m* indicates the number of threads created and dispatched during policy execution phase. Use the `mmcloudgateway` command configuration tuning settings to set your migrate or recall thread counts.

  Note: You must know the number of processors that are available on your Transparent cloud tiering service node.
- *-B* indicates the maximum number of files passed to each invocation of the EXEC script specified in the `<rules.file>`

Note: These two parameters (-m and -B) can be adjusted to improve the performance of large scale migrations.

The following sample policies are available in the package in the **/opt/ibm/MCStore/samples** folder:

Table 57. Sample policy list

No	Policy Name	Description
1	cloudDestroy.policy.template	Apply this policy for manually destroying orphaned cloud objects before retention time expires.
2	coresidentMigrate.template	Apply this policy for migrating files in the co-resident state, so that applications do not need to frequently recall files.
3	coResidenttoResident.template	Apply this policy if you want to convert all "co-resident" files in a file system to "resident".
4	listMigratedFiles.template	This policy will list all co-resident and resident files in the file system.
5	migrateFromDirectory.policy.template	This policy will migrate all files in a specified directory to the cloud storage tier.
6	recallFromCloud.policy.template	Apply this policy to recall files from the cloud storage tier.
7	thresholdBasedMigration.policy.template	Apply this policy to automatically migrate files from the file system to the cloud storage upon reaching certain threshold levels.
8	thumbnailTransparentRecall.policy.template	This policy will help you display the thumbnails when files are listed in tools such as Windows Explorer.
9	transparentRecall.policy.template	Transparent recall pulls files from the cloud when they are accessed by an application (read or write).

Migrating files to the cloud storage tier

This topic provides a brief description on how to migrate files to the cloud storage tier by using Transparent cloud tiering.

Note: Before you try to migrate files to the cloud storage tier, ensure that your cloud service configuration is completed as summarized in Chapter 6, "Configuring and tuning your system for Cloud services," on page 55.

You can trigger migration of files from your file system to an external cloud storage tier either transparently or manually. Transparent migration is based on the policies that are applied on a file system. Data is automatically moved from the system pool to the configured external storage tier when the system pool reaches a certain threshold level. A file can be automatically migrated to or from cloud storage pool based on some characteristics of the file such as age, size, last access time, path. Alternatively, the user can manually migrate specific files or file sets to a cloud storage pool. For more information on policy-based migration, see "Applying a policy on a Transparent cloud tiering node" on page 656.

To manually trigger migration of the file *file1*, issue this command:

```
mmcloudgateway files migrate file1
```

The state of the file becomes **Non-resident** after it is successfully migrated to the cloud storage tier.

If you want to migrate files in the co-resident state where the file has been copied to the cloud but also allows the data to be retained on the file system, see "Pre-migrating files to the cloud storage tier."

command. For more information on manually migrating files to the cloud storage tier, see `mmcloudgateway` command in the *IBM Spectrum Scale: Command and Programming Reference*.

Pre-migrating files to the cloud storage tier

Using Transparent cloud tiering, you can migrate files in both co-resident and non-resident states depending on the type of data (warm/cold).

Normally, when a file is migrated to the cloud storage tier, the status of the file becomes non-resident. This means that, you need to completely recall the file from the cloud to be able to perform any read or write operation. This might be an issue when the data is warm. The calling application must recall the file every time it needs to perform any operation on the file, and this can be resource-intensive. The solution to this issue is, migrating files in co-resident state or pre-migration.

In this type of migration, irrespective of the file size, when the files are warm, they are archived to the cloud storage in the co-resident state. That allows applications to have continued access to the files without issuing any recall commands, but at the same time, ensures that data is at least available on the cloud if there is any type of disaster. As data gets colder, the files are migrated in the non-resident state. Since files are available both in the file system and on the cloud, the storage utilization is more here.

Small files whose data resides entirely in the inodes are migrated in the co-resident state even if they are cold.

Note: When files are pushed through NFS that need to be moved immediately to the cloud tier, be aware that CES NFS keeps NFSv3 files open and keeps them open indefinitely for performance reasons. Any files that are cached in this manner will not be migrated to by Transparent cloud tiering to the cloud tier. NVSv4 client caching is more measured and less likely to prevent files from being migrated to the cloud tier and is recommended for this sort of usage.

You can migrate files in the co-resident state by using a policy as well as the CLI.

You can use the following command to migrate files in the co-resident state, where the --co-resident-state keyword is mandatory:

```
mmcloudgateway files migrate --co-resident-state file1
```

To verify that the file is migrated in the co-resident state, issue the following command:

```
mmcloudgateway files list file1
```

The system displays an output similar to this:

```
File name    : /gpfs0/file1
On-line size : 46
Used blocks  : 0
Data Version : 1
Meta Version : 1
State        : Co-resident
Base Name    : 57FA294111831B2B.10D9F57158C1628B.0063C1580F522F09.0000000000000000.5713748E.0000000000009E2B
```

For transparent migration in the co-resident state, the following policy has to be applied to the files by using the **mmchpolicy** command. A sample policy is available here: /opt/ibm/MCStore/samples/coresidentMigrate.template:

```
/*******************************************************************************
 * Licensed Materials - Property of IBM
 *
 *  OCO Source Materials
 *
 *  (C) Copyright IBM Corp. 2017 All Rights Reserved
 *
 *  The source code for this program is not published or other-
 *  wise divested of its trade secrets, irrespective of what has
 *  been deposited with the U.S. Copyright Office.
 *******************************************************************************/
define(
       exclude_list,
       (
        FALSE
        OR PATH_NAME LIKE '%/.mcstore/%'
        OR PATH_NAME LIKE '%/.mcstore.bak/%'
       )
)

/* Define premigrate pool, where files are migrated in co-resident state. This represent files moved
to cloud but also available locally on Scale file system.
* It is to be used for warmer data, as that data needs to be available locally on Scale file system
too, to avoid cloud round trips.
*/
RULE EXTERNAL POOL 'premigrate' EXEC '/usr/lpp/mmfs/bin/mmcloudgateway files' OPTS '--co-resident-state -F'

/* Define migrate pool, where files are migrated in non-resident state. This represent files are moved
to cloud and are not available locally.
 * It is to be used for colder data depending on file size. Larger colder files are made non-resident,
where as smaller files (less than 4K) are kept co-resident.
 */
RULE EXTERNAL POOL 'migrate' EXEC '/usr/lpp/mmfs/bin/mmcloudgateway files' OPTS '-F'

/* This rule defines movement of warm data. Each file (irrespective of it's size) is moved to cloud
in a co-resident state.
 * It means, file is available on the cloud and, access to it is possible from the hot-standby site
if needed.
 * Here the sample time interval to indicate warm data is, data that is not accessed between 10
to 30 days.
 * We don't want to pick up HOT data that is being accessed in last 10 days.
 * Another advantage of this co-resident migration is when data eventually gets colder, since it
is already migrated to cloud, only file truncation happens later.
 */
RULE 'MoveWarmData' MIGRATE FROM POOL 'system'
 THRESHOLD(0,0)
 TO POOL 'premigrate'
  WHERE NOT(exclude_list) AND
   (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '10' DAYS) AND
   (CURRENT_TIMESTAMP - ACCESS_TIME < INTERVAL '30' DAYS)


/* This rule defines movement of large files that are cold. Here, files that are above 4KB in size
are made non-resident to save
 * space on Scale file system. For files that are smaller than 4KB are anyway stored in inode
block itself.
```

```
 */
RULE 'MoveLargeColdData' MIGRATE FROM POOL 'system'
 THRESHOLD(0,0)
 TO POOL 'migrate'
  WHERE(KB_ALLOCATED > 4) AND NOT(exclude_list) AND
   (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '30' DAYS)

/* This rule defines movement of smaller files that are cold. Here, files that are less than
4KB in size are made co-resident, as
* there is no saving in moving these files, as data resides within the inode block, and not
on disk. It avoids un-necessary recall cycles.
*/
RULE 'MoveSmallColdData' MIGRATE FROM POOL 'system'
 THRESHOLD(0,0)
 TO POOL 'premigrate'
  WHERE(KB_ALLOCATED < 4) AND NOT(exclude_list) AND
   (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '30' DAYS)

/* Define default placement rule */
RULE 'Placement' SET POOL 'system'
```

Recalling files from the cloud storage tier

This topic provides a brief description on how to recall files from the cloud storage tier by using Transparent cloud tiering.

You can trigger recall of files from the cloud storage tier either transparently or manually. Transparent recall is based on the policies that are applied on a file system. Data is automatically moved from the cloud storage tier to the system pool when the system pool reaches a certain threshold level. A file can be automatically recalled from the cloud storage tier based on some characteristics of the file such as age, size, last access time, path. Alternatively, the user can manually recall specific files or file sets. For more information on policy-based recall, see "Applying a policy on a Transparent cloud tiering node" on page 656.

You can enable or disable transparent recall for a container when a container pair set is created. For more information, see "Binding your file system or fileset to the Cloud service by creating a container pair set" on page 61.

Note: Like recalls in IBM Spectrum Archive and IBM Spectrum Protect for Space Management (HSM), Transparent cloud tiering recall would be filling the file with uncompressed data and the user would need to re-compress it by using **mmrestripefs** or **mmrestripefile** if so desired. Since we are positioning compression feature for cold data currently, the fact of a file being recalled means the file is no longer cold and leaving the file uncompressed would allow better performance for active files.

To manually trigger recall of the file *file1*, issue this command:

```
mmcloudgateway files recall file1
```

The state of the file becomes **Co-resident** after it is successfully recalled. If the file that has been recalled no longer needs to be kept in the cloud tier it can be deleted. For more information on deleting a file in the co-resident state, see "Cleaning up files transferred to the cloud storage tier" on page 663.

For more information on manually recalling files, see `mmcloudgateway` command in the *IBM Spectrum Scale: Command and Programming Reference*.

Reconciling files between IBM Spectrum Scale file system and cloud storage tier

This topic describes how to reconcile files that are migrated between IBM Spectrum Scale file systems and the cloud tier.

Note: To run reconcile on a given Transparent cloud tiering managed file system, ensure that enough storage capacity is available temporarily under the root file system, to allow policy scan of a file system. Rough space requirements are (300 x *NF*), where *NF* is the number of files. For example, if a Transparent

cloud tiering managed file system has 1 billion inodes, then temporary space requirement for reconcile would be 300 GB (300 x 1 B). For more information, see the *-s LocalWorkDirectory* option in the *mmapplypolicy command* in the *IBM Spectrum Scale: Command and Programming Reference*.

The purpose of reconcile to is ensure that the cloud database is aligned properly with the IBM Spectrum Scale file system on state of files that have been tiered to the cloud. Such discrepancies can take place due to power outages and other such failures. It is recommended that this command be run every couple of months. This command needs to be run on every container pair. It should not be run in parallel with other maintenance commands like full cloud database backup but should be run in parallel with other maintenance commands (or migration policies) that affect that particular container. Also, this command should not be run while a policy migrate is being run.

There is another reason that you may want to run reconcile. Although there is a policy currently in place to automatically delete files in the cloud that have been deleted in the file system and similar support for older versions of files, that support is not fully guaranteed to remove a file. When for legal reasons or when there is a critical need to know for sure that a file has been deleted from the cloud, it is recommended that you run the reconcile command as shown below.

For example:

```
mmcloudgateway files reconcile --container-pair-set-name MyContainer gpfs_Container

Wed Nov 15 11:29:35 EST 2017
processing /ibm/gpfs_Container
Wed Nov 15 11:29:38 EST 2017  Reconcile started.
Wed Nov 15 11:29:38 EST 2017  Creating snapshot of the File System...
Wed Nov 15 11:29:39 EST 2017  Running policy on Snapshot to generate list of files to process.
Wed Nov 15 12:52:50 EST 2017  Removing snapshot.
Wed Nov 15 12:52:52 EST 2017  Reconcile is using a deletion retention period of 30 days.
Wed Nov 15 12:54:03 EST 2017  Reconcile will be processing 92617766 inode entries.
Wed Nov 15 12:54:03 EST 2017  Adding missing migrated files to the database...
Wed Nov 15 12:55:21 EST 2017  Processed 926178 entries out of 92617766.
Wed Nov 15 12:56:12 EST 2017  Processed 1852356 entries out of 92617766.
Wed Nov 15 12:56:59 EST 2017  Processed 2778533 entries out of 92617766.
Wed Nov 15 12:57:46 EST 2017  Processed 3704711 entries out of 92617766.
Wed Nov 15 12:58:34 EST 2017  Processed 4630889 entries out of 92617766.
Wed Nov 15 12:59:20 EST 2017  Processed 5557066 entries out of 92617766.

...
Wed Nov 15 14:13:15 EST 2017  Processed 92617766 entries out of 92617766.
Wed Nov 15 14:13:19 EST 2017  Reconcile found 228866 files that had been
migrated and were not in the directory.
Wed Nov 15 14:13:19 EST 2017  Reconcile detected 0 deleted files that were
deleted more than 30 days ago.
Wed Nov 15 14:13:19 EST 2017  Reconcile detected 12 migrated files that have
been deleted from the local file system, but have not been deleted from object
storage because they are waiting for their retention policy time to expire.
Wed Nov 15 14:13:19 EST 2017  Please use the 'mmcloudgateway files cloudList'
command to view the progress of the deletion of the cloud objects.
Wed Nov 15 14:13:21 EST 2017  Reconcile successfully finished.
mmcloudgateway: Command completed.
```

gpfs_Container is the device name of the file system that is associated with the node class, and **MyContainer** is the container where the cloud objects are stored.

You can delete files from cloud storage by using the deletion policy manager. However, you can also guarantee deletion by using a reconcile to manage the mandatory deletions. For example, if a migrated file is removed from the file system, a reconcile guarantees removal of the corresponding cloud objects and references that are contained in the cloud directory. Additionally, if multiple versions of a file are stored on the cloud, reconcile removes all older cloud versions (keeping the most recent). For example, if

a file is migrated, then updated, and migrated again. In this case, two versions of the file are stored on the cloud. Reconcile removes the older version from the cloud. Reconcile also deletes cloud objects that are no longer referenced.

Note: Reconcile removes entries from the cloud directory that references deleted file system objects. Therefore, it is recommended that you restore any files that must be restored before you run a reconcile. It is also recommended to run the reconciliation operation as a background activity during low load on the Transparent cloud tiering service nodes.

Cleaning up files transferred to the cloud storage tier

This topic describes how to clean up files that are transferred to the cloud storage tier and those have not yet been deleted from the IBM Spectrum Scale file system.

To clean up files on cloud storage that have already been deleted from IBM Spectrum Scale, see "Deleting cloud objects."

To do basic cleanup of objects that are transferred to the cloud object storage by using Transparent cloud tiering, issue a command according to this syntax:

```
mmcloudgateway files delete
 {-delete-local-file | -recall-cloud-file |
--require-local-file} [--keep-last-cloud-file]
 [--] File [File ...]
```

where,

- **--recall-cloud-file**: When this option is specified, the files are recalled from the cloud storage before deleting them on the cloud. The status of the local files becomes resident after the operation.
- **--delete-local-file**: This option deletes both local files and the corresponding cloud object. There is no recall here.
- **--keep-last-cloud-file**: This option deletes all the versions of the file except the last one from the cloud. For example, if a file has three versions on the cloud, then versions 1 and 2 are deleted and version 3 is retained.
- **--require-local-file**: This option removes the extended attribute from a co-resident file and makes it resident, without deleting the corresponding cloud objects. The option requires the file data to be present on the file system and will not work on a non-resident file.
- **--File**: This option can be used to process a file list similar to the one generated by the ILM policy.

The `mmcloudgateway files delete` command accepts files in GPFS file system as an input.

Deleting cloud objects

The standard way to delete cloud files is to set the `--cloud-retention-period-days` setting that sets a policy that indicates how long a file should be retained after it has been deleted from the file system before it should be deleted from the cloud storage tier. Periodically, the cloud directory is scanned for deleted files by a cloud destroy utility that runs in the background. That utility checks to see which files meet the criteria of having been deleted and retained longer than the cloud retention period. Files exceeding the cloud retention period days are deleted automatically by that background utility.

You can delete files by using `mmcloudgateway files delete` command or by using external commands such as `rm`. With any of these commands, the files are only deleted from the local file system, but the corresponding cloud objects are just marked for deletion. These marked objects are retained on the cloud for 30 days, by default. You can however modify the retention time by issuing the `mmcloudgateway config set` command. Once the retention period expires, the marked files are permanently deleted from the cloud storage tier.

It is recommended that you apply the destroy policy described below because of how file deletion works. For example, when you delete files by using external commands, the cloud objects are immediately

marked for deletion only if you have applied the destroy policy to the file system by using the **mmchpolicy** command. If the destroy policy is not applied, the cloud objects are marked for deletion only when you run the reconcile operation. The destroy policy is available here: `/opt/ibm/MCStore/samples/cloudDestroy.policy.template`. Additionally, it is recommended to apply the destroy policy along with other policies such as transparent recall and migration.

If you want to permanently delete the marked files before the retention time expires, you can use the **mmcloudgateway files destroy** command.

For example, to set the retention period of the cloud objects to 60 days, issue the following command:

```
mmcloudgateway config set --cloud-retention-period-days 60
```

You can permanently delete the cloud objects that are marked for deletion from the cloud automatically by using the destroy policy or the reconcile command - but you must do this only after 60 days of marking. If you want to delete these objects earlier than 60 days (for example, 30 days), specify the following command:

```
mmcloudgateway files destroy --cloud-retention-period-days 30 --container-pair-set-name container-1
    --filesystem-path /gpfs/myfold
```

Out of all the files marked for deletion, the command deletes cloud objects that were marked for deletion 30 days or earlier. The cloud objects that were marked for deletion less than 30 days are retained.

For more information on the destroy options, see the **mmcloudgateway** man page.

Listing files migrated to the cloud storage tier

Even if the files are deleted from the file system after migration, you can generate a list of files that are migrated to the cloud storage tier. By using the file names, you can use the **mmcloudgateway restore** option to retrieve the files back from the cloud storage tier.

To list the files that are migrated to the cloud, issue a command according to this syntax:

```
mmcloudgateway files cloudList {--path Path [--recursive [--depth Depth]] [--file File] |
                    --file-versions File |
                    --files-usage --path Path  [--depth Depth] |
                    --reconcile-status --path Path |
                    --path Path --start YYYY-MM-DD[-HH:mm] --end YYYY-MM-DD[-HH:mm]}
```

Note: You can specify `--reconcile-status` only if one reconcile is running at a time. (You can run multiple reconciles in parallel, but the progress indication has this limitation.)

For example, to list all files in the current directory, issue this command:

```
mmcloudgateway files cloudList --path /gpfs0/folder1
```

To list all files in all directories under the current directory, issue this command:

```
mmcloudgateway files cloudList --path /gpfs0/folder1 --recursive
```

To find all files named `myfile` in all directories under the current directory, issue this command:

```
mmcloudgateway files cloudList --path /gpfs0/folder1 --file myfile
```

To find all files named `myfile` in the current directory, issue this command:

```
mmcloudgateway files cloudList --path /gpfs0/folder1 --depth 0 --file myfile
```

To display information about all versions of file `myfile` in current directory, issue this command:

```
mmcloudgateway files cloudList --file-versions myfile
```

Restoring files

This topic provides a brief description on how to restore files that have been migrated to the cloud storage tier if the original files are deleted from the GFPS file system.

This option provides a non-optimized (emergency) support for manually restoring files that have been migrated to the cloud storage tier if the original stub files on the GPFS file system are deleted.

Note: Transparent cloud tiering does not save off the IBM Spectrum Scale directory and associated metadata such as ACLs. If you want to save off your directory structure, you need use something other than Transparent cloud tiering.

Before restoring files, you must identify and list the files that need to be restored by issuing the `mmcloudgateway files cloudList` command.

Assume that the file, `afile`, is deleted from the file system but is present on the cloud, and you want to find out what versions of this file are there on the cloud. To do so, issue the following command:

```
mmcloudgateway files cloudList --file-versions /gpfs0/afile
```

The system displays output similar to this:

```
id      datatime    datasize      metatime     metasize filename
13  Apr 27 03:34           6  Apr 27 03:34          499 /gpfs0/afile
14  Apr 27 03:35          12  Apr 27 03:35          693 /gpfs0/afile
```

You can use the output of the cloudList command for restoring files. For more information on the **cloudList** command, see "Listing files migrated to the cloud storage tier" on page 664.

To restore files, issue a command according to this syntax:

```
mmcloudgateway files restore [-v] [--overwrite] [--restore-stubs-only]
                            { -F FileListFile | [--dry-run] [--restore-location RestoreLocation]
                            [ --id ID] [--] File}
```

By using this command, you can restore files in two different ways. That is, the files to be restored along with their options can be either specified at the command line, or in a separate file provided by the -F option.

If you want to specify the options in a file, create a file with the following information (one option per line):

```
filename=<name of the file to be retrieved>
target=<full path to restore the file>
id=<This is the unique ID that is given to each version the file (This information is available
in the cloudList output>. If the id is not given, then the latest version of the file will
be retrieved.
```

The following example shows how the content needs to be provided in a file (for example, `filestoberestored`) for restoring a single file `/gpfs0/afile` with multiple versions:

```
# Restoring filename /gpfs0/afile
   filename=/gpfs0/afile
   target=/gpfs0/afile-33
   id=33
   %%
   filename=/gpfs0/afile
   target=/gpfs0/afile-34
   id=34
   %%
   # Restoring filename /gpfs0/afile
   filename=/gpfs0/afile
   target=/gpfs0/afile-35
   id=35
```

```
%%
# Restoring filename /gpfs0/afile
filename=/gpfs0/afile
target=/gpfs0/afile-latest
%%
# Restoring filename /gpfs0/afile
filename=/gpfs0/afile
```

The following example shows how the content needs to be provided in a file (for example,
filestoberestored) for restoring the latest version of multiple files (file1, file2, and file3):

```
# Restoring filename /gpfs0/file1, /gpf0/file1, and /gpfs0/file3
filename=/gpfs0/file1
target=/gpfs0/file1
%%
filename=/gpfs0/file2
target=/gpfs0/file2
%%
filename=/gpfs0/file3
target=/gpfs0/file3
```

Files to be restored are separated by lines with %% and # represents the comments.

Now that you have created a file with all required options, you need to pass this file as input to the
mmcloudgateway files restore command, as follows:

```
mmcloudgateway files restore -F filestoberestored
```

Note: It is advised not to run the delete policy if there is some doubt that the retention policy might
result in deleting of the file before you can restore it.
For information on the description of the parameters, see the **mmcloudgateway** command in *IBM Spectrum
Scale: Command and Programming Reference*.

Restoring Cloud services configuration

If your Cloud services configuration data is lost due to any unforeseen incident, you can restore the data
by using the **mmcloudgateway service restoreConfig** command.

To restore the configuration data and save it to the CCR, issue a command according to the following
syntax:

```
mmcloudgateway service restoreConfig --backup-config-file <name of the tar file>
```

For example, issue the following command to restore configuration data from the file,
tct_config_backup_20170915_085741.tar:

```
mmcloudgateway service restoreConfig --backup-config-file tct_config_backup_20170915_085741.tar
```

The system displays an output similar to this:

```
You are about to restore the TCT Configuration settings to the CCR.
Any new settings since the backup was made will be lost.
The TCT servers should be stopped prior to this operation.

Do you want to continue and restore the TCT cluster configuration?
Enter "yes" to continue: yes
mmcloudgateway: Unpacking the backup config tar file...
mmcloudgateway: Completed unpacking the tar file.

   Restoring the Files:
     [_cloudnodeclass1.settings  -  Restored]
     [_cloudnodeclass.settings  -  Restored]
     [mmcloudgateway.conf  -  Restored]

mmcloudgateway: TCT Config files were restored to the CCR.
mmcloudgateway: Command completed.
```

Note: During the restore operation, Transparent cloud tiering servers are restarted on all the nodes.

Checking the Cloud services database integrity

There might be situations where the Cloud services database that is associated with a container gets corrupted secondary to a power outage or system crash, and in such cases, you must check the integrity of the database before you proceed with any operation.

To check the integrity of database that is associated with a container, issue a command according to the following syntax:

```
mmcloudgateway files checkDB --container-pair-set-name ContainerPairSetName
```

For example, issue the following command to check the integrity of the database that is associated with the container, `container1`:

```
mmcloudgateway files checkDB --container-pair-set-name container1
```

The system displays output similar to this:

```
CheckDB returned OK.
mmcloudgateway: Command completed.
```

If the database is corrupted, the system displays an output similar to this:

```
MCSTG000130E: Command Failed with following reason: Request failed with error :
Database is potentially not in a good state and needs to be rebuilt.
mmcloudgateway: Command failed. Examine previous error messages to determine cause.
```

Note: Cloud services configurations contain sensitive security-related information regarding encryption credentials, so you must store your configuration back-up in a secure location. This configuration information is critical in helping you restore your system data, if necessary.

Restoring Transparent cloud tiering service on a backup cluster

Transparent cloud tiering cluster can go down because of any critical hardware/disk failures or some other unforeseen events. This topic describes the procedure for restoring a Transparent cloud tiering service on a back-up cluster.

Before you begin, ensure the following:
- The file system names of the back-up cluster (cluster 2) are identical to the file system names of the original cluster (cluster 1).
- The Transparent cloud tiering node group name of cluster 2 is same as that of cluster 1.

Do the following the steps:
1. Install the Transparent cloud tiering server rpm on the cluster 2 nodes.
2. Copy the Transparent cloud tiering backup configuration file of the original cluster (cluster 1) to a desired location in cluster 2.
3. Restore the configuration data to the CCR of cluster 2 by using a command similar to this:

   ```
   mmcloudgateway service restoreConfig --backup-file tct_backup_file_20170919_130910.tar
   ```

   In this example, `tct_backup_file_20170919_130910.tar` is the name of the backup configuration file of cluster 1.
4. Start the Transparent cloud tiering service on cluster 2:

   ```
   mmcloudgateway service start -N NodeClass_cluster2
   ```
5. Restore the database that is associated with each container:

   ```
   mmcloudgateway files rebuilddb Container1 /gpfs0
   ```
6. Ensure that all files that are migrated from cluster 1 are listed:

```
|    mmcloudgateway files CloudList
```
| 7. Restore all files that are migrated to the cloud storage:

```
|    mmcloudgateway files restore
```

| **Note:** This command restores all files that are passed to it in a single thread sequentially (in a series),
| which can be time-consuming if there are many files. When you must restore many files, do it in
| parallel. See "Restoring files" on page 665 for more information.

Manual recovery of Transparent cloud tiering database

Transparent cloud tiering uses a database called "cloud directory" to store a list and versions of files that
are migrated to the cloud. Any issues in this database might lead to undesired results. If the database is
corrupted or has been accidentally deleted, you can reconstruct it from automatic backups of this
database that are kept on the cloud.

You need to perform a recovery of the database when Transparent cloud tiering produces any of the
following messages in the logs:
- The cloud directory database for file system /dev/gpfs0 could not be found. Manual recovery is
  necessary.
- The directory service for file system /dev/gpfs0 is not ready for use. Manual recovery is necessary.
- The cloud directory database for file system /dev/gpfs0 is corrupted. Manual recovery is necessary.

To perform a manual recovery, issue a command according to this syntax:

```
| mmcloudgateway files rebuildDB --container-pair-set-name ContainerPairSetName Device
```

where,

`filesystem` is the device name of the file system whose database is corrupted and which is in need of
manual recovery.

| `--container-pair-set-name` is the name of the container associated with the file system or fileset.

| For example, if you want to recover the database associated with the file system, /dev/gpfs0 and the
| container, container-1, issue this command:

```
| mmcloudgateway files rebuildDB --container-pair-set container-1 /dev/gpfs0
```

| The system displays output similar to this:

```
| mmcloudgateway: Command completed.
```

Cloud data sharing

You can share data between storage servers by using the import and export function available in IBM
Spectrum Scale.

Cloud data sharing works by combining the import and export functions that allow data to be moved
across disparate geographical locations and/or heterogeneous application platforms. Cloud data sharing
maintains a set of records of those moves called a manifest that enable applications to know what has
moved. An application at one site can generate data, export it to the cloud, and applications at other sites
can import and process that data. Applications can know what data has moved and is, therefore, now
available by looking at the manifest file. It is also a way to easily move data back and forth between local
and cloud storage systems. Cloud data sharing supports moving data to the cloud, and pulling data from
the cloud . Cloud data sharing must be configured with a local file system and a cloud account. Once
configured, data can be moved between the IBM Spectrum Scale file system and the cloud account.

Application considerations

Exporting applications need some mechanism to both notify other applications that new data is available on the cloud and give those applications some way of understanding what objects were put to the cloud. Cloud data sharing services provide a manifest to help applications communicate that new data is available and what that data is. When data is exported, an option to build a manifest file can be specified. This manifest is a text file that contains the name of the cloud objects exported and some other information that can be used by an application that wants to import the full data, or a subsection of it.

When data is imported, there are cases in which not all the data is needed and this unneeded data can be identified by information in the file metadata. In these cases, it is recommended that as a first pass the file headers are imported only with the "import-only-stub" option. The policy engine can then be used to import only those files that are needed, thereby saving transfer time and cost. For now this import of stub includes metadata only for data that was previously exported by IBM Spectrum Scale.

Note: For many cloud services, enabling indexed containers can impact performance, so it is possible that cloud containers are not indexed. For these situations, a manifest is mandatory. But even with indexing enabled, for large containers that contain many objects, a manifest can be useful.

Additionally, this manifest utility can be used by a non-Spectrum Scale application to build a manifest file for other applications, including IBM Spectrum Scale, to use for importing purposes.

There is a manifest utility that can run separate from IBM Spectrum Scale (it is a Python script) that can be used to look at the manifest. It provides a way to list and filter the manifest content, providing comma separated value output.

An overview on using import and export CLI commands

To export files to a cloud storage tier, issue a command according to the following syntax:
```
mmcloudgateway files export
    [--tag Tag]
    [--target-name TargetName]
    [--container Container | no-container ]
    [--manifest-file MainifestFile]
    [--export-metadata [--fail-if-metadata-too-big]]
    [--strip-filesystem-root]
    File[ File ] }
```

The following example exports a local file named /dir1/dir2/file1 to the cloud and store it in a container named "MyContainer". A manifest file will be created, and the object exported to the cloud will have an entry in that manifest file tagged with "MRI_Images".
```
mmcloudgateway files export --container MyContainer --tag MRI_Images --export-metadata --manifest-file
/dir/ManifestFile /dir1/dir2/file1
```

To import files from a cloud storage tier, issue a command according to the following syntax:
```
mmcloudgateway files import
    [--container Container | no-container ]
    [--import-only-stub]
    [--import-metadata ]
    { [--directory Directory] | [--directory-root DirectoryRoot] | [--target-name TargetName] }
    { PolicyFile -e | [--] File[ File ] }
```

The following example imports files from the cloud storage tier and creates a necessary local directory structure.
```
mmcloudgateway files import --directory /localdir /dir1/dir2/file1
```

For more information on the usage of the import and export functions, see the **mmcloudgateway** man page.

Listing files exported to the cloud

This topic describes how to parse a manifest file and how to list files from the cloud.

Although files are exported to the cloud from a IBM Spectrum Scale environment, the files can be imported by a non-IBM Spectrum Scale application. While you export files to the cloud, a manifest file is built. The manifest file includes a list of these exported files and the metadata associated with native object storage.

When data is exported to the cloud, the manifest file is not automatically pushed to the cloud. You must decide when and where to export the manifest file.

When to transfer: If you are using a policy to export data, a good time to export the manifest is immediately after the policy has successfully executed your executive chain. Waiting too long can result in manifest that is too big and that does not provide frequent enough guidance to applications looking for notifications about new data on the cloud. Constantly pushing out new manifests can create other problems where the applications have to deal with many small manifests, and having to understand which they should use.

Where to transfer: Unlike transparent cloud tiering, cloud data sharing allows data to be transferred to any container at any time. This freedom can be very useful, especially when setting up multiple tenants. A centralized manifest is useful in a single tenant environment, but when there are multiple tenants with different access privileges to different files it may be better to split up your manifest destinations accordingly. Export all data targeted to a particular tenant and then send the manifest. Export data for the next tenant, and so forth.

The manifest file is a text file whose entry format is as follows:
```
<File/Object Name> <CloudContainerName> <TagID> <TimeStamp><Newline>
```

Typically, this file is not accessed directly but rather is accessed using the manifest utility.

A manifest utility produces a CSV stream entry format is as follows:
```
<TagID>,<CloudContainerName>,<TimeStamp>,<File/Object Name><newline>
```

where,
- `TagID` is an optional identifier the object is associated with.
- `CloudContainerName` is the name of the container the object was exported into.
- `TimeStamp` follows the format : "DD MON YYYY HH:MM:SS GMT".
- `File/Object Name` can contain commas, but not new line characters.

An example entry in a manifest utility stream output is as follows:
```
0, imagecontainer, 6 Sep 2016 20:31:45 GMT, images/a/cat.scan
```

You can use the **mmcloudmanifest** tool to parse the manifest file that is created by the **mmcloudgateway files export** command or by any other means. By looking at the manifest files, an application can download the desired files from the cloud.

The **mmcloudmanifest** tool is automatically installed on your cluster along with Transparent cloud tiering rpms. However, you must install the following packages for the tool to work:
- Install Python version 2.7.5
- Install pip. For more information, see https://packaging.python.org/install_requirements_linux/
- Install `apache-libcloud` package by running the **sudo pip install apache-libcloud** command.

The syntax of the tool is as follows:

```
mmcloudmanifest
ManifestName [--cloud --properties-file PropertiesFile --manifest-container ManifestContainer
[--persist-path PersistPath]]
[--tag-filter TagFilter] [--container-filter ContainerFilter]
[--from-time FromTime] [--path-filter PathFilter]
[--help]
```

where,

- `ManifestName`: Specifies the name of the manifest object that is there on the cloud. For using a local manifest file, specify the full path name to the manifest file.
- `--properties-file PropertiesFile`: Specifies the location of the properties file to be used when retrieving the manifest file from the cloud. A template properties file is located at `/opt/ibm/MCStore/scripts/provider.properties`. This file includes details such as the name of the cloud storage provider, credentials, and URL.
- `--persist-path PersistPath`: Stores a local copy of the manifest file that is retrieved from the cloud in the specified location.
- `--manifest-container ManifestContainer`: Name of the container in which the manifest is located.
- `--tag-filter TagFilter`: Lists only the entries whose Tag ID # matches the specified regular expression (regex).
- `--container-filter ContainerFilter`: Lists only the entries whose container name matches the specified regex.
- `--from-time FromTime`: Lists only the entries that occur starting at or after the specified time stamp. The time stamp must be enclosed within quotations, and it must be in the 'DD MON YYYY HH:MM:SS GMT' format. Example: '21 Aug 2016 06:23:59 GMT'
- `--path-filter PathFilter`: Lists only the entries whose path name matches the specified regex.

The following command exports four CSV files tagged with "us-weather", along with the manifest file, "manifest.txt", to the cloud:

```
mmcloudgateway files export --container arn87817249811111500553 --manifest-file manifest.txt
--tag us-weather /gpfs/weather_data/MetData_Oct06-2016-Oct07-2016-ALL.csv
/gpfs/weather_data/MetData_Oct07-2016-Oct08-2016-ALL.csv
/gpfs/weather_data/MetData_Oct08-2016-Oct09-2016-ALL.csv
/gpfs/weather_data/MetData_Oct09-2016-Oct10-2016-ALL.csv
/gpfs/weather_data/MetData_Oct10-2016-Oct11-2016-ALL.csv
```

The following command exports four CSV files tagged with "uk-weather", along with the manifest file, "manifest.text", to the cloud:

```
mmcloudgateway files export --container arn87817249811111500553 --manifest-file manifest.txt
--tag uk-weather /gpfs/weather_data/MetData_Oct06-2016-Oct07-2016-ALL.csv
/gpfs/weather_data/MetData_Oct07-2016-Oct08-2016-ALL.csv
/gpfs/weather_data/MetData_Oct08-2016-Oct09-2016-ALL.csv
/gpfs/weather_data/MetData_Oct09-2016-Oct10-2016-ALL.csv
/gpfs/weather_data/MetData_Oct10-2016-Oct11-2016-ALL.csv
```

So, the container "arn87817249811111500553" contains both US and UK weather data.

The following command parses the manifest file and imports the files that are tagged with "us-weather" to the local file system under the `/gpfs` directory:

```
mmcloudmanifest parse-manifest manifest.txt --tag-filter us-weather
 | xargs mmcloudgateway files import --directory /gpfs --container arun87817249811111500553
```

You can verify these files by using the following command:

```
ls -l /gpfs
```

The system displays output similar to this:

```
total 64
drwxr-xr-x. 2 root root  4096 Oct  5 07:09 automountdir
-rw-r--r--. 1 root root  7859 Oct 18 02:15 MetData_Oct06-2016-Oct07-2016-ALL.csv
-rw-r--r--. 1 root root  7859 Oct 18 02:15 MetData_Oct07-2016-Oct08-2016-ALL.csv
-rw-r--r--. 1 root root 14461 Oct 18 02:15 MetData_Oct08-2016-Oct09-2016-ALL.csv
-rw-r--r--. 1 root root 14382 Oct 18 02:15 MetData_Oct09-2016-Oct10-2016-ALL.csv
-rw-r--r--. 1 root root 14504 Oct 18 02:15 MetData_Oct10-2016-Oct11-2016-ALL.csv
drwxr-xr-x. 2 root root  4096 Oct 17 14:12 weather_data
```

Importing cloud objects exported through an old version of Cloud data sharing

There might be situations where you exported files to the cloud storage tier by using Cloud services 4.2.3.x (before upgrade) and then want to import those files by using Cloud services 5.0.0 (after upgrade). This topic describes the procedure for importing such files.

1. If the container includes migrated files, clean up the associated fileset or file system objects from the container.

2. Delete the container pair set associated with the container in the new version of Transparent cloud tiering by using the `mmcloudgateway containerPairSet delete` command.

3. Create a Cloud data sharing service by using the `mmcloudgateway cloudService create` command.

4. Using the sharing service that is created in the previous step, create a container pair set pointing to the same container name as the container in the previous version of Transparent cloud tiering:

   ```
   mmcloudgateway containerPairSet create
   ```

   Note: If encryption was enabled in the older release, then you must include these parameters while creating a container pair set:
   - KeyManagerName
   - ActiveKey

5. Import the files by using the `mmcloudgateway files import` command.

Scheduling recommendations for Transparent cloud tiering tasks

This topic provides recommendations for scheduling different tasks provided in the table.

- The following work is best scheduled to be run with no overlap with migration policy runs or when recall traffic is lighter for that node group.

- Distribute scheduled batch work in time slots so that activities do not overlap within the node group (between node groups is fine).

- If you decide to schedule some tasks in parallel, it is best to distribute where possible the work on different nodes since these activities can take up a lot of CPU and memory resource.

- After you have set it up, measure how long tasks take to ensure a good distribution of the tasks
  - Activity durations can grow as the number of files tiered or the file system size grows. You may need to adjust your schedule occasionally to adjust for the longer task times

Table 58. Schedule recommendations

Task	Frequency	Granularity	Parallelism allowed
File migration by policy. Add reference link here	As needed	Per Policy	By file system/ file sent that the policy is scoped to
Reconcile. Add reference link here	Once a month*	Per tiering container pair	By file system/ file sent that the policy is scoped to
Clean up deleted files. Add reference link here	Once a week*	Per tiering container pair	By file system/ file sent that the policy is scoped to
Backup full cloud database. Add reference link here	Once a week*	Per tiering container pair	By container pair

Table 58. Schedule recommendations (continued)

Task	Frequency	Granularity	Parallelism allowed
Backup configuration. Add reference link here	Whenever configuration changes	Per node class	By node class

Note: In case of a cold archive where a container pair is no longer changing (no deletions and migrations) then frequency can be reduced significantly.

Known limitations of Cloud services

This topic describes the limitations that are identified for Cloud services.

`mmcloudgateway files migrate *` on a parent folder does not move all files within the subfolders
> Running the **`mmcloudgateway files migrate`** command to migrate all files (including the files within the subfolders) does not migrate all files within subfolders. It only migrates leaf files within the current folder, from which the migrate command is issued. The migrate process skips the subfolders, by displaying the following warning message:
>
> ```
> MCSTG00051E: File is not a regular file. Migration requests only support regular files.
> error processing /<file-system-mount>/<folder1>/<folder-2>....
> ```
>
> To migrate all files (including files within the subfolders) in one go, issue this command:
>
> ```
> find <gpfs-mountpoint-folder-or-subfolder> -type f —exec mmcloudgateway files migrate {} +
> ```
>
> This command passes the entire list of files to a single migrate process in the background as follows:
>
> ```
> mmcloudgateway files migrate <file1> <file2> <sub-folder1/file1> <sub-folder2/file1> ......
> ```

Migrating Transparent cloud tiering specific configuration to cloud storage might lead to issues
> While you move data to an external cloud storage tier, it is required not to migrate files within the Transparent cloud tiering internal folder (.mcstore folder within the configured GPFS file system) to cloud storage. It might lead to undesirable behavior for the Transparent cloud tiering service. To address this issue, include the EXCLUDE directive in the migration policy.
>
> Refer to the `/opt/ibm/MCStore/samples` folder to view sample policies that can be customized as per your environment and applied on the file system that is managed by Transparent cloud tiering.

Running `mmcloudgateway files delete` on multiple files
> Trying to remove multiple files in one go with the **`mmcloudgateway files delete delete-local-file`** command fails with a **NullPointerException**. This happens while you clean up the cloud metrics. Issue this command to remove the cloud objects:
>
> ```
> find <gpfs-file-system> -type f -exec mmcloudgateway files delete {} \;
> ```

Range reads from the cloud object storage is not supported for transparent recall.
> When a file is transparently recalled, the file is entirely recalled.

Policy-based migrations
> Policy-based migrations should be started only from Transparent cloud tiering server nodes. Client nodes should be used only for manual migration.

File names with carriage returns or non-UTF-8 characters
> Transparent cloud tiering does not perform any migration or recall operation on files whose names include carriage returns or non-UTF-8 characters.

File systems mounted with the `nodev` option
> If a file system is mounted with the `nodev` option, then it cannot be mounted to a directory with an existing folder with the same name as the file system. Transparent cloud tiering is not supported in this situation.

Administrator cannot add a container pair set while managing a file system with 'automount' setting turned on.
> Make sure that `automount` setting is not turned on while a file system is in use with Transparent cloud tiering.

Files created through NFS clients when migrated to the cloud storage tier

> If caching is turned on on the NFS clients (with the **--noac** option) while mounting the file system, files that are migrated to the cloud storage tier remain in the co-resident status, instead of the non-resident status.

Transparent cloud tiering configured with proxy servers
> IBM Security Key Lifecycle Manager does not work when Transparent cloud tiering is configured with proxy servers.

Transparent cloud tiering supports Swift Dynamic Large Objects only. Ensure that below configuration is in included in the Swift "required_filters" section, as follows:

```
/usr/lib/python2.7/site-packages/swift/proxy/server.py

required_filters = [
    {'name': 'catch_errors'},
    {'name': 'gatekeeper',
     'after_fn': lambda pipe: (['catch_errors']
                               if pipe.startswith('catch_errors')
                               else [])},
    {'name': 'dlo', 'after_fn': lambda _junk: [
        'copy', 'staticweb', 'tempauth', 'keystoneauth',
        'catch_errors', 'gatekeeper', 'proxy_logging']},
    {'name': 'versioned_writes', 'after_fn': lambda _junk: [
        'slo', 'dlo', 'copy', 'staticweb', 'tempauth',
        'keystoneauth', 'catch_errors', 'gatekeeper', 'proxy_logging']},
    # Put copy before dlo, slo and versioned_writes
    {'name': 'copy', 'after_fn': lambda _junk: [
        'staticweb', 'tempauth', 'keystoneauth',
        'catch_errors', 'gatekeeper', 'proxy_logging']}]
```

No support for file systems earlier than 4.2.x
> Cloud services support IBM Spectrum Scale file systems versions 4.2.x and later only.

Running reconciliation during heavy writes and reads on the file system
> Reconciliation fails when it is run during heavy I/O operations on the file system.

For current limitations and restrictions, see IBM Spectrum Scale FAQs.

For more information, see the topic *Interoperability of Transparent Cloud Tiering with other IBM Spectrum Scale features* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Chapter 39. Managing file audit logging

The following topics describe various ways to manage file audit logging in IBM Spectrum Scale.

For more information about file audit logging, see *Introduction to file audit logging* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Stopping consumers in file audit logging

The user can stop all consumers on file systems that are being audited if necessary.

To stop consumers on a given set of nodes in IBM Spectrum Scale, issue a command similar to the following example:

```
mmaudit all consumerStop -N Node1,[Node2,...]
```

For more information, see *Consumers in file audit logging* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*, "Starting consumers in file audit logging," *Monitoring the consumer status* in *IBM Spectrum Scale: Problem Determination Guide*, and *mmaudit command* in *IBM Spectrum Scale: Command and Programming Reference*.

Starting consumers in file audit logging

The user can start all consumers on file systems that are being audited if necessary.

To start consumers on a given set of nodes in IBM Spectrum Scale, issue a command similar to the following example:

```
mmaudit all consumerStart -N Node1,[Node2,...]
```

For more information, see *Consumers in file audit logging* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*, "Stopping consumers in file audit logging," *Monitoring the consumer status* in *IBM Spectrum Scale: Problem Determination Guide*, and the *mmaudit command* in *IBM Spectrum Scale: Command and Programming Reference*.

Displaying topics that are registered in the message queue for file audit logging

The user can run the **mmmsgqueue list --topics** command to see a list of topics for file audit logging.

Run the following command to display topics in the message queue:

```
mmmsgqueue list --topics
```

If there are file audit logging topics in the message queue, they will display an output like the following example:

```
157_6372129557625143312_24_audit
```

In the example output, the 6372129557625143312 number is the GPFS cluster ID. This number can help distinguish the topic from other topics if the user is working in an environment with multiple clusters.

The inclusion of **audit** at the end of the name means that the file is registered for file audit logging. For more information, see *The message queue in file audit logging* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Enabling file audit logging on a new spectrumscale cluster node

Use this information to configure file audit logging on a node that was added to a **spectrumscale** cluster.

For more information about adding nodes to an existing installation, see *Adding nodes, NSDs, or file systems to an existing installation* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Adding nodes to the message queue after it has been enabled is not supported. To enable file audit logging on a new **spectrumscale** cluster node, you must disable file audit logging on all file systems where it is enabled, remove the message queue configuration, add the message queue configuration with the new broker nodes specified, and then re-enable file audit logging on the file systems. The following set of steps details the process:

1. Issue the following command to view the file systems that are enabled for file audit logging:

   mmaudit all list

2. Issue the following command to disable file audit logging on all file systems that have it enabled:

   mmaudit Device disable

3. Reissue the following command. Verify that you get the following message when file audit logging has been disabled on all file systems:

   ```
   # mmaudit all list
   [I] File audit logging is disabled for all devices.
   ```

4. Issue the following command to disable the message queue:

   mmmsgqueue config --remove

   Note: This will also remove the message queue node classes and configuration information:

5. Issue the following command and verify that you get the following message:

   ```
   # mmmsgqueue status
   [I] MsgQueue currently not enabled.
   ```

6. Ensure that all rpm, package, OS, and hardware requirements stated in the *Requirements and limitations for file audit logging* topic in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* are met by the new node.

   Note: Software requirements can be installed and verified using the installation toolkit. A node can be added through the toolkit or by manually installing the required rpm and packages using the package installation command based on the OS.

7. Enable file audit logging. For more information, see "Enabling file audit logging on a file system" on page 83.

   Note: Remember to add the new node when you enable the message queue.

8. Verify the nodes that are running the processes and their current states. For more information, see *Monitoring the message queue server and Zookeeper status* in *IBM Spectrum Scale: Problem Determination Guide*.

Chapter 40. Highly available write cache (HAWC)

Highly available write cache (HAWC) reduces the latency of small write requests by initially hardening data in a non-volatile fast storage device prior to writing it back to the backend storage system.

Overview and benefits

Current disk drive systems are optimized for large streaming writes, but many workloads such as VMs and databases consist of many small write requests, which do not perform well with disk drive systems. To improve the performance of small writes, storage controllers buffer write requests in non-volatile memory before writing them to storage. This works well for some workloads, but the amount of NVRAM is typically quite small and can therefore not scale to large workloads.

The goal of HAWC is to improve the efficiency of small write requests by absorbing them in any nonvolatile fast storage device such as SSDs, Flash-backed DIMMs, or Flash DIMMs. Once the dirty data is hardened, GPFS can immediately respond to the application write request, greatly reducing write latency. GPFS can then flush the dirty data to the backend storage in the background.

By first buffering write requests, HAWC allows small writes to be gathered into larger chunks in the page pool before they are written back to storage. This has the potential to improve performance as well by increasing the average amount of data that GPFS writes back to disk at a time.

Further, when GPFS writes a data range smaller than a full block size to a block for the first time, the block must first be fully initialized. Without HAWC, GPFS does this by writing zeroes to the block at the time of the first write request. This increases the write latency since a small write request was converted into a large write request (for example, a 4K write request turns into a 1MB write request). With HAWC, this initialization can be delayed until after GPFS responds to the write request, or simply avoided altogether if the application subsequently writes the entire block.

To buffer the dirty data, HAWC hardens write data in the GPFS recovery log. This means that with HAWC, the recovery log must be stored on a fast storage device because if the storage device on which the recovery log resides is the same as the data device, HAWC will decrease performance by writing data twice to the same device. By hardening data in the recovery log, all incoming requests are transformed into sequential operations to the log. In addition, it is important to note that applications never read data from the recovery log, since all data that is hardened in the recovery log is always kept in the page pool. The dirty data in the log is only accessed during file system recovery due to improper shutdown of one or more mounted instances of a GPFS file system.

The maximum size of an individual write that can be placed in HAWC is currently limited to 64KB. This limit has been set for several reasons, including the following:

- The benefit of writing data to fast storage decreases as the request size increases.
- Fast storage is typically limited to a much smaller capacity than disk subsystems.
- Each GPFS recovery log is currently limited to 1GB. Every file system and client pair has a unique recovery log. This means that for each file system, the size of HAWC scales linearly with every additional GPFS client. For example, with 2 file systems and 10 clients, there would be 20 recovery logs used by HAWC to harden data.

Note that combining the use of HAWC with LROC allows GPFS to leverage fast storage on application reads and writes.

Applications that can benefit from HAWC

Typically, it is recommended to place GPFS metadata in a storage pool consisting of fast storage devices such as SSDs. Storing GPFS recovery logs in fast storage improves the performance of metadata-intensive workloads where the recovery log is heavily used and when GPFS is configured to replicate data.

With HAWC, storing the recovery log in fast storage has the added benefit that workloads that experience bursts of small and synchronous write requests (no matter if they are random or sequential) will also be hardened in the fast storage. Well-known applications that exhibit this type of write behavior include VMs, databases, and log generation.

Since the characteristics of fast storage vary greatly, users should evaluate their application workload with HAWC in their storage configuration to ensure a benefit is achieved. In general, however, speedups should be seen in any environment that either currently lacks fast storage or has very limited (and non-scalable) amounts of fast storage.

Restrictions and tuning recommendations for HAWC

When enabling HAWC, take the following restrictions and tuning recommendations into consideration:

Ping pong recovery log buffers
> Ping pong recovery log buffers should not be enabled when the recovery log is stored on storage devices that can gracefully write data upon power failure. This restriction includes SSDs, NVRAM, storage controllers, and RAID controllers among others.
>
> Ping pong buffers are only needed to avoid data corruption when the recovery log is stored directly on disk. They place log data in two separate locations on disk to avoid loss of that data if a sector becomes unavailable. Writing to two separate locations creates additional overhead that is exacerbated by HAWC due to the large amount of data it places in the recovery log.
>
> In general, it is not recommended to use HAWC with any storage device that requires ping pong buffers to be enabled because it doubles the amount of data that must be written before GPFS can respond to an application write request.
>
> To disable log buffers, run the following command:
> ```
> mmchconfig logPingPongSector=no
> ```

Recovery log size
> The size of the recovery log defaults to a small value (less than 16 MB), which is not sufficient space to buffer HAWC data. Therefore, it is recommended to increase the size of the log at least to 128 MB or larger (1 GB maximum). However, the effect of a larger recovery log is that upon node failure, more data must be recovered into the storage system, which increases the time that it takes to recover. It is important to take this factor into account because applications will not be able to access data in the file system while recovery is running.

Encryption
> Encrypted data is never stored in the recovery log, but instead follows the pre-GPFS 4.1.0.4 semantics for synchronous writes even if the HAWC threshold is set to a value greater than 0.

Small files and directory blocks
> HAWC does not change the following behaviors:
> - write behavior of small files when the data is placed in the inode itself
> - write behavior of directory blocks or other metadata

Using HAWC

Learn how to enable HAWC, set up storage for the recovery log, and do administrative tasks.

"Enabling HAWC"

"Setting up the recovery log in fast storage"

"Administrative tasks"

Enabling HAWC

To enable HAWC, set the write cache threshold for the file system to a value that is a multiple of 4 KB and in the range 4 KB - 64 KB. The following example shows how to set the threshold for an existing file system:

```
mmchfs gpfsA --write-cache-threshold 32K
```

The following example shows how to specify the threshold when you create a new file system:

```
mmcrfs /dev/gpfsB -F ./deiskdef2.txt -B1M --write-cache-threshold 32K -T /gpfs/gpfsB
```

After HAWC is enabled, all synchronous write requests less than or equal to the write cache threshold are put into the recovery log. The file system sends a response to the application after it puts the write request in the log. If the size of the synchronous write request is greater than the threshold, the data is written directly to the primary storage system in the usual way.

Setting up the recovery log in fast storage

Proper storage for the recovery log is important to improve the performance of small synchronous writes and to ensure that written data survives node or disk failures. Two methods are available:

Method 1: Centralized fast storage
> In this method, the recovery log is stored on a centralized fast storage device such as a storage controller with SSDs, a flash system, or an IBM Elastic Storage™ Server (ESS) with SSDs.
>
> You can use this configuration on any storage that contains the system pool or the system.log pool. The faster that the metadata pool is compared to the data storage, the more using HAWC can help.

Method 2: Distributed fast storage in client nodes
> In this method, the recovery log is stored on IBM Spectrum Scale client nodes on local fast storage devices, such as SSDs, NVRAM, or other flash devices.
>
> The local device NSDs must be in the system.log storage pool. The system.log storage pool contains only the recovery logs.
>
> It is a good idea to enable at least two replicas of the system.log pool. Local storage in an IBM Spectrum Scale node is not highly available, because a node failure makes the storage device inaccessible.
>
> Use the **mmchfs** command with the **--log-replicas** parameter to specify a replication factor for the **system.log** pool. This parameter, with the **system.log** capability, is intended to place log files in a separate pool with replication different from other metadata in the system pool.
>
> You can change log replication dynamically by running the **mmchfs** command followed by the **mmrestripefs** command. However, you can enable log replication only if the file system was created with a number of maximum metadata replicas of 2 or 3. (See the **-M** option of the **mmcrfs** command .)

Administrative tasks

Learn how to do the following administrative tasks with HAWC:

Restriping after you add or remove a disk

As with any other pool, after you add or remove a disk from the `system.log` pool, run the **mmrestripefs -b** command to rebalance the pool.

Preparing for a node or disk failure in the `system.log` pool

- If the `system.log` is replicated, you can run the following command to ensure that data is replicated automatically after a node or disk fails:

  `mmchconfig restripeOnDiskFailure=yes -i`

- You can run the following command to set how long the file system waits to start a restripe after a node or disk failure:

  `mmchconfig metadataDiskWaitTimeForRecovery=Seconds`

  where *Seconds* is the number of seconds to wait. This setting helps to avoid doing a restripe after a temporary outage such as a node rebooting. The default time is 300 seconds.

Adding HAWC to an existing file system

Follow these steps:

1. If the metadata pool is not on a fast storage device, migrate the pool to a fast storage device. For more information, see "Managing storage pools" on page 358.
2. Increase the size of the recovery log to at least 128 MB. Enter the following command:

   mmchfs Device -L *LogFileSize*

   where *LogFileSize* is the size of the recovery log. For more information, see the topic *mmchfs command* in the *IBM Spectrum Scale: Command and Programming Reference* guide.
3. Enable HAWC by setting the write cache threshold, as described earlier in this topic.

Chapter 41. Local read-only cache

Many applications benefit greatly from large local caches. Not only is the data available with very low latency, but the cache hit serves to reduce the load on the shared network and on the backend storage itself, thus benefiting all nodes, even those without large caches.

Local solid-state disks (SSDs) provide an economical way to create very large caches. The SSD cache serves as an extension to the local buffer pool. As user data or metadata is evicted from the buffer pool in memory, it can be stored in the local cache. A subsequent access will retrieve the data from the local cache, rather than from the home location. The data stored in the local cache, like data stored in memory, remains consistent. If a conflicting access occurs, the data is invalidated from all caches. In a like manner, if a node is restarted, all data stored in the cache is discarded.

In theory, any data or metadata can be stored in the local SSD cache, but the cache works best for small random reads where latency is a primary concern. Since the local cache typically offers less bandwidth than the backend storage, it might be unsuitable for large sequential reads. The configuration options provide controls over what is stored in the cache. The default settings are targeted at small random I/O.

The local read-only cache (LROC) function is disabled by default. To enable it, the administrator must define an NSD for an LROC device. The LROC device is expected to be a solid-state disk (SSD) accessible via SCSI. The device is defined as a standard NSD by **mmcrnsd**, but the **DiskUsage** is set to **localCache**. The NSD must have a primary server and is not allowed to have other servers. The primary server must be the node where the physical LROC device is installed. The device is *not* exported to other nodes in the cluster. The storage pool and failure group defined for the NSD are ignored and should be set to null. The **mmcrnsd** command writes a unique NSD volume ID onto the device.

The minimum size of a local read-only cache device is 4 GB. The local read-only cache requires memory equal to 1% of the capacity of the LROC device.

Once the LROC device is defined, the daemon code at the primary server node is automatically told to do device discovery. The daemon detects that **localCache** is defined for its use and determines the mapping to the local device. The daemon then informs the local read-only cache code to begin using the device for caching. Currently, there is a limit of four **localCache** devices per node. Note that the daemon code does not need to be restarted to begin using the cache.

The LROC device can be deleted by using the **mmdelnsd** command. Both **mmcrnsd** and **mmdelnsd** can be issued while the daemon is running with file systems mounted and online. The call to delete the NSD first informs the daemon that the device is being deleted, which removes it from the list of active LROC devices. Any data cached on the device is immediately lost, but data cached on other local LROC devices is unaffected. Once the **mmdelnsd** command completes, the underlying SSD can be physically removed from the node.

The NSD name for the LROC device cannot be used in any other GPFS commands, such as **mmcrfs**, **mmadddisk**, **mmrpldisk**, **mmchdisk** or **mmchnsd**. The device is shown by **mmlsnsd** as a **localCache**.

Note: To avoid a security exposure, by default IBM Spectrum Scale does not allow file data from encrypted files, which is held in memory as cleartext, to be copied into an LROC device. However, you can set IBM Spectrum Scale to allow cleartext from encrypted files to be copied into an LROC device with the following command:

```
mmchconfig lrocEnableStoringClearText=yes
```

You might choose this option if you have configured your system to remove the security exposure.

| **Warning:** If you allow cleartext from an encrypted file to be copied into an LROC device, you must take
| steps to protect the cleartext while it is in LROC storage.
| For more information, see the following links:
| "Encryption and a local read-only cache (LROC) device" on page 638
| The topic *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Chapter 42. Miscellaneous advanced administration topics

The following topics provide information about miscellaneous advanced administration tasks:

- "Changing IP addresses and host names"
- "Enabling a cluster for IPv6" on page 684
- "Using multiple token servers" on page 684
- "Exporting file system definitions between clusters" on page 685
- "IBM Spectrum Scale port usage" on page 685

Changing IP addresses and host names

GPFS assumes that IP addresses and host names remain constant. In the rare event that such a change becomes necessary or is inadvertently introduced by reinstalling a node with a disk image from a different node for example, follow the steps in this topic.

If all of the nodes in the cluster are affected:

1. Issue the **mmshutdown -a** command to stop GPFS on all nodes.
2. Using the documented procedures for the operating system, add the new host names or IP addresses but do not remove the old ones yet. This can be achieved, for example, by creating temporary alias entries in **/etc/hosts**. Do not restart the nodes until the **mmchnode** command in Step 3 is executed successfully. If any of these conditions cannot be met, use the alternate procedure described in this section.
3. Update the nodes that the cluster uses as the administration interface node and the daemon interface node, if necessary. To update these values, issue the **mmchnode** command with the **--admin-interface** and the **--daemon-interface** options.

   Note: You cannot specify the **--daemon-interface** option for a quorum node if CCR is enabled. Temporarily change the node to a nonquorum node. Then issue the **mmchnode** command with the **--daemon-interface** option against the nonquorum node. Finally, change the node back into a quorum node.
4. If the IP addresses over which the subnet attribute is defined are changed, you must update your configuration by issuing the **mmchconfig** command with the **subnets** attribute.
5. Start GPFS on all nodes with **mmstartup -a**.
6. Remove the unneeded old host names and IP addresses.

If only a subset of the nodes are affected, it may be easier to make the changes using these steps:

1. Before any of the host names or IP addresses are changed:
   - Use the **mmshutdown** command to stop GPFS on all affected nodes.
   - If the host names or IP addresses of the primary or secondary GPFS cluster configuration server nodes must change, use the **mmchcluster** command to specify another node to serve as the primary or secondary GPFS cluster configuration server.
   - If the host names or IP addresses of an NSD server node must change, temporarily remove the node from being a server with the **mmchnsd** command. Then, after the node has been added back to the cluster, use the **mmchnsd** command to change the NSDs to their original configuration. Use the **mmlsnsd** command to obtain the NSD server node names.
   - Use the **mmdelnode** command to delete all affected nodes from the GPFS cluster.
2. Change the node names and IP addresses using the documented procedures for the operating system.

3. If the IP addresses over which the subnet attribute is defined are changed, you need to update your configuration by using the **mmchconfig** command with the **subnets** attribute.

4. Issue the **mmaddnode** command to restore the nodes to the GPFS cluster.

5. If necessary, use the **mmchcluster** and **mmchnsd** commands to restore the original configuration and the NSD servers.

Enabling a cluster for IPv6

For newly created clusters, if any of the specified node interfaces on the **mmcrcluster** command resolves to an IPv6 address, the cluster is automatically enabled for IPv6. For existing IPv4-based clusters, follow the applicable procedure described in this section.

If you are performing the procedure during a scheduled maintenance window and GPFS can be shut down on all of the nodes in the cluster, issue the command:

```
mmchconfig enableIPv6=yes
```

After the command finishes successfully, you can start adding new nodes with IPv6 addresses.

If it is not possible to shut down GPFS on all of the nodes at the same time, issue the command:

```
mmchconfig enableIPv6=prepare
```

The next step is to restart GPFS on each of the nodes so that they can pick up the new configuration setting. This can be done one node at a time when it is convenient. To verify that a particular node has been refreshed, issue:

```
mmdiag --config | grep enableIPv6
```

The reported value should be 1.

Once all of the nodes have been recycled in this manner, issue the command:

```
mmchconfig enableIPv6=commit
```

This command will only succeed when all GPFS daemons have been refreshed. Once this operation succeeds, you can start adding new nodes with IPv6 addresses.

To convert an existing node from an IPv4 to an IPv6 interface, use one of the procedures described in "Changing IP addresses and host names" on page 683.

Using multiple token servers

Distributed locking, allowing GPFS to maintain a consistent view of the file system, is implemented using token-based lock management. Associated with every lockable object is a token.

Before a lock on an object can be granted to a thread on a particular node, the lock manager on that node must obtain a token from the token server. The total number of token manager nodes depends on the number of manager nodes defined in the cluster.

When a file system is first mounted, the file system manager is the only token server for the file system. Once the number of external mounts exceeds one, the file system manager appoints all the other manager nodes defined in the cluster to share the token server load. Once the token state has been distributed, it remains distributed until all external mounts have gone away. The only nodes that are eligible to become token manager nodes are those designated as manager nodes.

The number of files for which tokens can be retained on a manager node is restricted by the values of the **maxFilesToCache** and **maxStatCache** configuration parameters of the **mmchconfig** command. Distributing

the tokens across multiple token manager nodes allows more tokens to be managed or retained concurrently, improving performance in situations where many lockable objects are accessed concurrently.

Exporting file system definitions between clusters

You can export a GPFS file system definition from one GPFS cluster to another.

To export file system definitions between clusters, follow these steps:

1. Ensure that all disks in all GPFS file systems to be migrated are in working order by issuing the **mmlsdisk** command. Verify that the disk status is ready and availability is up. If not, correct any problems and reissue the **mmlsdisk** command before continuing.
2. Stop all user activity in the file systems.
3. Follow any local administrative backup procedures to provide for protection of your file system data in the event of a failure.
4. Cleanly unmount all affected GPFS file systems. Do not use force unmount.
5. Export the GPFS file system definitions by issuing the **mmexportfs** command. This command creates the configuration output file *ExportDataFile* with all relevant file system and disk information. Retain this file as it is required when issuing the **mmimportfs** command to import your file systems into the new cluster. Depending on whether you are exporting a single file system or all of the file systems in the cluster, issue:

   ```
   mmexportfs fileSystemName -o ExportDataFile
   ```

   or

   ```
   mmexportfs all -o ExportDataFile
   ```
6. Ensure that the file system disks from the old GPFS cluster are properly connected, and are online and available to be accessed from appropriate nodes of the new GPFS cluster.
7. To complete the movement of your file systems to the new cluster using the configuration file created in Step 5, issue one of these commands, depending on whether you are importing a single file system or all of the file systems in the cluster:

   ```
   mmimportfs fileSystemName -i ExportDataFile
   ```

   or

   ```
   mmimportfs all -i ExportDataFile
   ```

IBM Spectrum Scale port usage

The nodes in a IBM Spectrum Scale cluster communicate with each other using the TCP/IP protocol. The port number used by the main GPFS daemon (**mmfsd**) is controlled with the **tscTcpPort** configuration parameter. The default port number is 1191.

You can specify a different port number using the **mmchconfig** command:

```
mmchconfig tscTcpPort=PortNumber
```

When the main GPFS daemon (**mmfsd**) is not running on the primary and backup configuration server nodes, a separate service (**mmsdrserv**) is used to provide access to the configuration data to the rest of the nodes in the cluster. The port number used for this purpose is controlled with the **mmsdrservPort** parameter. By default, **mmsdrserv** uses the same port number as the one assigned to the main GPFS daemon. If you change the daemon port number, you must specify the same port number for **mmsdrserv** using the following command:

```
mmchconfig mmsdrservPort=PortNumber
```

Do not change the **mmsdrserv** port number to a number different from that of the daemon port number.

Certain commands (**mmadddisk**, **mmchmgr**, and so on) require an additional socket to be created for the duration of the command. The port numbers assigned to these temporary sockets are controlled with the **tscCmdPortRange** configuration parameter. If an explicit range is not specified, the port number is dynamically assigned by the operating system from the range of ephemeral port numbers. If you want to restrict the range of ports used by IBM Spectrum Scale commands, use the **mmchconfig** command:

```
mmchconfig tscCmdPortRange=LowNumber-HighNumber
```

In a remote cluster setup, if IBM Spectrum Scale on the remote cluster is configured to use a port number other than the default, you have to specify the port number to be used with the **mmremotecluster** command:

```
mmremotecluster update ClusterName -n tcpPort=PortNumber,Node,Node...
```

Table 59 provides IBM Spectrum Scale port usage information:

Table 59. IBM Spectrum Scale port usage

Descriptor	Explanation
Service provider	IBM Spectrum Scale
Service name	**mmfsd** **mmsdrserv**
Port number	1191 While executing certain commands, IBM Spectrum Scale may need to create additional sockets whose dynamic port numbers are assigned by the operating system. Such sockets are used by commands to exchange data with GPFS daemons running on other nodes. The port numbers that are used correspond to the ephemeral ports of the operating system. To control which ports are used by the commands (so that firewall rules can be written to allow incoming traffic only on those ports), you can restrict the port range to a specific range by setting the **tscCmdPortRange** configuration variable.
Protocols	TCP/IP
Source port range	The source port range is chosen by the operating system on the client side.
Is the service name/number pair in the default /etc/services file shipped with AIX and Linux distributions?	See the IBM Spectrum Scale FAQ in IBM Knowledge Center(www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
Is the service name/number pair added to /etc/services by a product?	No
Binaries that listen on the ports	/usr/lpp/mmfs/bin/mmfsd /usr/lpp/mmfs/bin/mmsdrserv

Table 59. IBM Spectrum Scale port usage (continued)

Descriptor	Explanation

Can the service be configured to use a different port?	Yes. To change the main port used by IBM Spectrum Scale, enter:
	
	`mmchconfig tscTcpPort=PortNumber`
	
	Note: If you change the main port (daemon port) number, you must change the **mmsdrserv** port to the same number.
	
	To change the **mmsdrserv** port number to match the daemon port number, use:
	
	`mmchconfig mmsdrservPort=PortNumber`
	
	To change the range of port numbers used for command execution, use:
	
	`mmchconfig tscCmdPortRange=LowNumber-HighNumber`
	
	To specify a port number when connecting to remote clusters, use the **mmremotecluster** command.

When is the service required? What depends on the service?	On the IBM Spectrum Scale primary and secondary cluster configuration servers, either **mmsdrserv** or **mmfsd** needs to be running at all times to provide access to IBM Spectrum Scale configuration data to the rest of the cluster. On other nodes, **mmfsd** must be running in order to mount a IBM Spectrum Scale file system. Depending on the IBM Spectrum Scale configuration, a node either has to be a member of the IBM Spectrum Scale cluster or possess an authorized SSL key in order to establish a connection.
When the daemon starts and its port is already in use (for example, another resource has bound to it already), how does the daemon behave?	The daemon shuts down and tries to start over again. Most GPFS daemon down error messages are in the **mmfs.log.previous** log for the instance that failed. If the daemon restarted, it generates a new **mmfs.log.latest** log. Begin problem determination for these errors by examining the operating system error log. IBM Spectrum Scale records file system or disk failures using the error logging facility provided by the operating system: **syslog** facility on Linux and **errpt** facility on AIX. See the *IBM Spectrum Scale: Problem Determination Guide* for further information.
Is there an administrator interface to query the daemon and have it report its port number?	Yes; issue this command: **`mmlsconfig tscTcpPort`**

Is the service/port registered with the Internet Assigned Numbers Authority (IANA)?	Yes
	
	`gpfs 1191/tcp General Parallel File System`
	`gpfs 1191/udp General Parallel File System`
	`# Dave Craft <gpfs@ibm.com>`
	`November 2004`

Note: Ports configured for the IBM Spectrum Scale remote shell command (such as `ssh`) or the remote file copy command (such as `scp`) and the ICMP echo command (network ping) also must be unblocked in the firewall for IBM Spectrum Scale to function properly.

Securing the IBM Spectrum Scale system using firewall

The IBM Spectrum Scale system is an open system where the customer can interact with the system through other third-party interfaces like MMC, web applications, and so on. The customer also has root access to the system just like any Linux server administrator. Firewalls that are associated with open systems are specific to deployments, operating systems, and it varies from customer to customer. It is the responsibility of the system administrator or Lab Service (LBS) to set the firewall accordingly; similar to what Linux distributions do today. This section provides recommendations to set up a firewall to secure the IBM Spectrum Scale protocol nodes.

Table 60. Firewall related information

Function	Firewall recommendations and considerations
IBM Spectrum Scale installation	"Firewall recommendations for the IBM Spectrum Scale installation"
Internal communication	"Firewall recommendations for internal communication among nodes" on page 689 For detailed information on port usage, see "IBM Spectrum Scale port usage" on page 685.
Protocol access (NFS, SMB, and Object)	"Firewall recommendations for protocol access" on page 690
IBM Spectrum Scale GUI	"Firewall recommendations for IBM Spectrum Scale GUI" on page 694
File encryption with IBM Security Key Lifecycle Manager (SKLM)	"Firewall recommendations for IBM SKLM" on page 695
File encryption with Vormetric Data Security Manager (DSM)	"Firewall recommendations for Vormetric DSM" on page 695
REST API	"Firewall recommendations for the REST API" on page 696
Performance monitoring	"Firewall recommendations for Performance Monitoring tool" on page 696
Active File Management (AFM)	"Firewall considerations for Active File Management (AFM)" on page 697
Transparent cloud tiering	*Firewall recommendations for Transparent cloud tiering* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
Remotely mounted file systems	"Firewall considerations for remote mounting of file systems" on page 697
IBM Spectrum Protect with IBM Spectrum Scale	"Firewall recommendations for using IBM Spectrum Protect with IBM Spectrum Scale" on page 698
IBM Spectrum Archive with IBM Spectrum Scale	"Firewall considerations for using IBM Spectrum Archive with IBM Spectrum Scale" on page 698
File audit logging	"Firewall recommendations for file audit logging" on page 698
Examples of opening firewall ports	

Firewall recommendations for the IBM Spectrum Scale installation

It is recommended to allow connection only from the IBM Spectrum Scale cluster node IPs (internal IPs and protocol IPs) on port 8889 and block all other external connections on this port during the installation process.

The installation toolkit uses the following ports during IBM Spectrum Scale installation.

Table 61. Recommended port numbers that can be used for installation

Port Number	Protocol	Service Name	Components involved in communication
8889	TCP	Chef	Intra-cluster and installer server
10080	TCP	Repository	Intra-cluster and installer server
123	UDP	NTP	Intra-cluster or external depending on the NTP server location

The port that is used during the installation (8889) can be blocked when the installation is over. You can get the list of protocol IPs by using the `mmlscluster --ces` command. Use the `mmlscluster` command to get the list of all internal IPs.

Chef is the underlying technology used by the installation toolkit. During installation, a Chef server is started on the installation server, and repositories are created to store information about the various IBM Spectrum Scale components. Each node being installed by the installation toolkit must be able to establish a connection to the repository and the Chef server itself. Typically, the installation toolkit is run on the intra-cluster network from a single node to all other nodes. An alternate method is available in which the installation server is designated as a node or server outside of the cluster with full access to at least one of the cluster nodes. The installation toolkit coordinates the installation from this external location, passing all necessary commands to the internal cluster node for it to actively assist with installation of the rest of the cluster. In this case, port 8889 and 10080 must be opened both to the intra-cluster network on all nodes and on the external client-facing network.

NTP is not necessary but time sync among nodes is highly recommended and it is required for protocol nodes.

Firewall recommendations for internal communication among nodes

The IBM Spectrum Scale system uses the following ports for internal communication among various IBM Spectrum Scale nodes.

Table 62. Recommended port numbers that can be used for internal communication

Port Number	Protocol	Service Name	Components that are involved in communication
1191	TCP	GPFS	Intra-cluster
22	TCP	SSH	Clients who are accessing the system
User selected range	TCP	GPFS ephemeral port range	Intra-cluster

- The SSH port 22 is used for command execution and general node-to-node configuration as well as administrative access.
- The primary GPFS daemons (`mmfsd` and `mmsdrserv`), by default, listen on port 1191. This port is essential for basic cluster operation. The port can be changed manually by setting the *mmsdrservPort* configuration variable with the `mmchconfig mmsdrservPort=`*PortNumber* command.
- The ephemeral port range of the underlying operating system is used when IBM Spectrum Scale creates additional sockets to exchange data among nodes. This occurs while executing certain commands and this process is dynamic based on the point in time needs of the command as well as

other concurrent cluster activities. You can define an ephemeral port range manually by setting the *tscCmdPortRange* configuration variable with the `mmchconfig` *tscCmdPortRange=LowNumber-HighNumber* command.

If the installation toolkit is used, the ephemeral port range is automatically set to 60000-61000. Firewall ports must be opened according to the defined ephemeral port range. If commands such as `mmlsmgr` and `mmcrfs` hang, it indicates that the ephemeral port range is improperly configured.

The following are the recommendations for securing internal communications among IBM Spectrum Scale nodes:

- Allow connection only to the GPFS cluster node IPs (internal IPs and protocol node IPs) on port 1191. Block all other external connections on this port. Use the `mmlscluster --ces` command to get the list of protocol node IP and use the `mmlscluster` command to get the list of IPs of internal nodes.
- Allow all external communications request that are coming from the admin or management network and IBM Spectrum Scale internal IPs on port 22.
- Certain commands such as `mmadddisk`, `mmchmgr`, and so on require an extra socket to be created for the duration of the command. The port numbers that are assigned to these temporary sockets are controlled with the **tscCmdPortRange** configuration parameter. If an explicit range is not specified, the port number is dynamically assigned by the operating system from the range of ephemeral port numbers. It is highly recommended to set the port range. For more information on how to set the port range, see "IBM Spectrum Scale port usage" on page 685.

Firewall recommendations for protocol access

It is recommended to use certain port numbers to secure the protocol data transfer.

Recommendations for NFS access

The following table provides the list of static ports that are used for NFS data I/O.

Table 63. Recommended port numbers for NFS access

Port Number	Protocol	Service Name	Components that are involved in communication
2049	TCP and UDP	NFSV4 or NFSV3	NFS clients and IBM Spectrum Scale protocol node
111	TCP and UDP	RPC (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	STATD (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	MNT (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	NLM (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	RQUOTA (required by both NFSV3 and NFSV4)	NFS clients and IBM Spectrum Scale protocol node

Note: NFSV3 uses the dynamic ports for NLM, MNT, and STATD services. When an NFSV4 server is used with the firewall, these services must be configured with static ports.

The following recommendations are applicable:

- Review your `systems/etc/services` file in order to select the static ports to use for MNT, NLM, STATD, and RQUOTA services that are required by the NFSV4 server. Do not use a port that is already used by another application. Set the static ports by using the **mmnfs config change** command. Allow TCP and UDP port 2049 to use the protocol node IPs. For example:

  `mmnfs config change MNT_PORT=32767:NLM_PORT=32769:RQUOTA_PORT=32768:STATD_PORT=32765`

- Allow all external communications on TCP and UDP port 111 by using the protocol node IPs.
- Allow all external communications on the TCP and UDP port that is specified with **mmnfs config change** for MNT and NLM ports.
- Ensure that following steps are done after making any of these changes.
  - Restart NFS after changing these parameters by using the following commands.

    ```
    mmces service stop NFS -a
    mmces service start NFS -a
    ```

  - Use **rpcinfo -p** to query the protocol nodes after any port changes to verify that proper ports are in use.
  - Remount any existing clients because a port change might have disrupted connections.

Recommendations for SMB access

Samba uses the following ports for the secure access.

Table 64. Recommended port numbers for SMB access

Port Number	Protocol	Service Name	Components that are involved in communication
445	TCP	Samba	SMB clients and IBM Spectrum Scale protocol node
4379	TCP	CTDB	Inter-protocol node

The following recommendations are applicable for the SMB access:

- Allow the access request that is coming from the data network and admin and management network on port 445 using the protocol node IPs. You can get the list of protocol node IPs by using the **mmlscluster --ces** command.
- Allow connection only to the requests that are coming from the IBM Spectrum Scale cluster node IPs (internal IPs and protocol node IPs) on port 4379. Block all other external connections on this port. Use the **mmlscluster** command to get the list of cluster node IPs.

Port usage for BLOCK service

Table 65. Recommended port numbers for iSCSI access

Port Number	Protocol	Service Name	Components that are involved in communication
3260	TCP	BLOCK (iSCSI)	IBM Spectrum Scale protocol node (when the BLOCK service is enabled) listening on this port

Object port configuration

Note: IBM Spectrum Scale is configured with the ports listed here. Changing ports requires updating configuration files, Keystone endpoint definitions, and SELinux rules. This must be done only after careful planning.

The following table lists the ports configured for object access.

Table 66. Port numbers for object access

Port Number	Protocol	Service Name	Components that are involved in communication
8080	TCP	Object Storage Proxy	Object clients and IBM Spectrum Scale protocol node
6200	TCP	Object Storage (local account server)	Local host
6201	TCP	Object Storage (local container server)	Local host
6202	TCP	Object Storage (local object server)	Local host
6203	TCP	Object Storage (object server for unified file and object access)	Local host
11211	TCP and UDP	Memcached (local)	Local host

The following ports are configured for securing object access:
* Allow all external communications on TCP port 8080 (Object Storage proxy).
* Allow connection only from the IBM Spectrum Scale cluster node IPs (internal IPs and protocol node IPs) on ports 6200, 6201, 6202, 6203, and 11211. Block all other external connections on this port.

Shell access by non-root users must be restricted on IBM Spectrum Scale protocol nodes where the object services are running to prevent unauthorized access to object data.

Note: The reason for these restrictions is that because there is no authentication of requests made on ports 6200, 6201, 6202, and 6203, it is critical to ensure that these ports are protected from access by unauthorized clients.

Port usage for object authentication

You can configure either an external or internal Keystone server to manage the authentication requests. Keystone uses the following ports:

Table 67. Port numbers for object authentication

Port Number	Protocol	Service Name	Components that are involved in communication
5000	TCP	Keystone Public	Authentication clients and object clients
35357	TCP	Keystone Internal/Admin	Authentication and object clients and Keystone administrator

These ports are applicable only if keystone is hosted internally on the IBM Spectrum Scale system. The following port usage is applicable:

- Allow all external communication requests that are coming from the admin or management network and IBM Spectrum Scale internal IPs on port 35357.
- Allow all external communication requests that are coming from clients to IBM Spectrum Scale for object storage on port 5000. Block all other external connections on this port.

Port usage to connect to the Postgres database for object protocol

The Postgres database server for object protocol is configured to use the following port:

Table 68. Port numbers for Postgres database for object protocol

Port Number	Protocol	Service Name	Components that are involved in communication
5431	TCP and UDP	postgresql-obj	Inter-protocol nodes

It is recommended to allow connection only from Cluster node IPs (Internal IPs and Protocol node IPs) on port 5431. Block all other communication requests on this port.

Note: The Postgres instance used by the object protocol uses port 5431. This is different from the default port to avoid conflict with other Postgres instances that might be on the system including the instance for IBM Spectrum Scale GUI.

Consolidated list of recommended ports that are used for installation, internal communication, and protocol access

The following table provides a consolidated list of recommended ports and firewall rules.

Table 69. Consolidated list of recommended ports for different functions

Function	Dependent network service names	External ports that are used for file and object access	Internal ports that are used for inter-cluster communication	UDP / TCP	Nodes for which the rules are applicable
Installer	Chef	N/A	8889 (chef) 10080 (repo)	TCP	GPFS server, NSD server, protocol nodes
GPFS (internal communication)	GPFS	N/A	1191 (GPFS) 60000-61000 for tscCmdPortRange 22 for SSH	TCP and UDP TCP only for 22	GPFS server, NSD server, protocol nodes
SMB	gpfs-smb.service gpfs-ctdb.service rpc.statd	445	4379 (CTDB)	TCP	Protocol nodes only

Table 69. Consolidated list of recommended ports for different functions (continued)

Function	Dependent network service names	External ports that are used for file and object access	Internal ports that are used for inter-cluster communication	UDP / TCP	Nodes for which the rules are applicable
NFS	ganesha.nfsd rpcbind rpc.statd	2049 (NFS_PORT - required only by NFSV3) 111 (RPC - required only by NFSV3) 32765 (STATD_PORT) 32767 (MNT_PORT - required only by NFSV3) 32768 (RQUOTA_PORT - required by both NFSV3 and NFSV4) 32769 (NLM_PORT - required only by NFSV3) **Note:** Make the dynamic ports static with command `mmnfs config change`.	N/A	TCP and UDP	Protocol nodes only
Object	swift-proxy-server keystone-all postgresql-obj	8080 (proxy server) 35357 (keystone) 5000 (keystone public)	5431 (Object Postgres instance) 6200-6203 (Object Storage) 11211 (Memcached)	TCP TCP and UDP (for 11211 only)	Protocol nodes only

Firewall recommendations for IBM Spectrum Scale GUI

Dedicating certain ports for firewalls helps to secure IBM Spectrum Scale management and installation GUIs. Different ports are used for securing installation GUI and management GUI.

The following table lists the ports that need to be used to secure GUI.

Table 70. Firewall recommendations for GUI

Port Number	Functions	Protocol
9080	Installation GUI	HTTP
9443	Installation GUI	HTTPS
80	Management GUI IBM Spectrum Scale management API	HTTP

Table 70. Firewall recommendations for GUI (continued)

Port Number	Functions	Protocol
443	Management GUI IBM Spectrum Scale management API	HTTPS
4444	Management GUI	Localhost only

All nodes of the IBM Spectrum Scale cluster must be able to communicate with the GUI nodes through the ports 80 and 443. If multiple GUI nodes are available in a cluster, the communication among those GUI nodes is carried out through the port 443.

Both the management GUI and IBM Spectrum Scale management API share the same ports. That is, 80 and 443. However, for APIs, the ports 443 and 80 are internally forwarded to 47443 and 47080 respectively. This is done automatically by an iptables rule that is added during the installation.

The management GUI uses ZIMon to collect performance data. ZIMon collectors are normally deployed with the management GUI and sometimes on other systems in a federated configuration. Each ZIMon collector uses three ports, which can be configured in ZIMonCollector.cfg. The default ports are 4739, 9085, and 9084.

The port 4444 is accessible only from the localhost.

Firewall recommendations for IBM SKLM

Read this topic to learn about port access for IBM Security Key Lifecycle Manager (SKLM).

The following table lists the ports for communicating with SKLM. The SKLM ports apply for both IBM Spectrum Scale file encryption and Transparent Cloud Tiering (TCT).

Table 71. Firewall recommendations for SKLM

Port Number	Protocol	Service	Components
• 9083	TCP	WebSphere Application Server	mmsklmconfig command for retrieving server certificate chain
• SKLM 2.6: 9080 • SKLM 2.7: 443	TCP	SKLM REST admin interface	mmsklmconfig utility for configuring Spectrum Scale
• 5696	TCP	SKLM Key Management Interoperability Protocol (KMIP) interface	Spectrum Scale daemon for retrieving encryption keys, mmsklmconfig utility for configuring Spectrum Scale

Firewall recommendations for Vormetric DSM

Read this topic to learn about port access for Vormetric Data Security Manager (DSM).

The following table lists the ports for communicating with DSM. The DSM ports are used by IBM Spectrum Scale file encryption.

Table 72. Firewall recommendations for SKLM

Port Number	Protocol	Service	Components
8445	TCP	DSM administration web GUI	The mmsklmconfig command for retrieving a server certificate chain

Table 72. Firewall recommendations for SKLM (continued)

Port Number	Protocol	Service	Components
5696	TCP	DSM Key Management Interoperability Protocol (KMIP) interface	The Spectrum Scale daemon for retrieving encryption keys

Firewall recommendations for the REST API

IBM Spectrum Scale REST API servers are accessible through port 8191.

The following table lists the ports for communicating with a REST API server.

Table 73. Firewall recommendations for REST API

Port Number	Interface type	Protocol
8191	REST API	HTTPS

Users of the REST API must communicate with a server through port 8191.

Firewall recommendations for Performance Monitoring tool

The IBM Spectrum Scale system uses the following ports for the Performance Monitoring tool to work.

Table 74. Recommended port numbers that can be used for Performance Monitoring tool

Port Number	Protocol	Service Name	Components that are involved in communication
4739	TCP and UDP	Performance Monitoring tool	Intra-cluster. This port needs to be accessible by all sensor nodes that are sending performance monitoring data.
8123	TCP	Object Metric collection	Intra-cluster
8124	TCP	Object Metric collection	Intra-cluster
8125	UDP	Object Metric collection	Intra-cluster
8126	TCP	Object Metric collection	Intra-cluster
8127	TCP	Object Metric collection	Intra-cluster
9084	TCP	Performance Monitoring Tool	Starting from IBM Spectrum Scale version 5.0.0, this port is limited to the loopback network interface by default. If the performance data should be available to other nodes, bind the query interface to 0.0.0.0, and limit access of this port to those nodes that should be able to access the collector's query interface.

Table 74. Recommended port numbers that can be used for Performance Monitoring tool (continued)

Port Number	Protocol	Service Name	Components that are involved in communication
9085	TCP	Performance Monitoring Tool	Intra-cluster. This port needs to be open to all nodes where the performance collector is running. This port is used for internal communication between the collectors.

Important:

*   The 4739 port needs to be open when a collector is installed.
*   The 9085 port needs to be open when there are two or more collectors.
*   If the 9084 port is closed, accessing the collector to debug or to connect external tools or, even another instance of the GUI, remotely is not possible, except from the node where the GUI and the collector are installed.

Firewall considerations for Active File Management (AFM)

Active File Management (AFM) allows one or more IBM Spectrum Scale clusters, or a non-IBM Spectrum Scale NFS source, to exchange file data. File data exchange between clusters can accomplish many goals, one of which is to allow for disaster recovery.

For AFM data transfers, either NFS or NSD is used as the transport protocol.
*   For port requirements of NFS, see "Firewall recommendations for protocol access" on page 690.
*   For port requirements of NSD, see "Firewall recommendations for internal communication among nodes" on page 689.

Firewall considerations for remote mounting of file systems

IBM Spectrum Scale clusters can access file systems on other IBM Spectrum Scale clusters using remote mounts.

Remote mounts can be used in the following ways.
*   All nodes in the IBM Spectrum Scale cluster requiring access to another cluster's file system must have a physical connection to the disks containing file system data. This is typically done through a storage area network (SAN).
*   All nodes in the IBM Spectrum Scale cluster requiring access to another cluster's file system must have a virtual connection through an NSD server.

In both cases, all nodes in the cluster requiring access to another cluster's file system must be able to open a TCP/IP connection to every node in the other cluster. For information on the basic GPFS cluster operation port requirements, see "Firewall recommendations for internal communication among nodes" on page 689.

Note: Each cluster participating in a remote mount might reside on the same internal network or on a separate network from the host cluster. From a firewall standpoint, this means that the host cluster might need ports to be opened to a number of external networks, depending on how many separate clusters are accessing the host.

Firewall recommendations for using IBM Spectrum Protect with IBM Spectrum Scale

The IBM Spectrum Scale **mmbackup** command is used to back up file systems and filesets to an externally located IBM Spectrum Protect server. IBM Spectrum Protect for Space Management (HSM) is used with the IBM Spectrum Scale policy engine to migrate data to secondary storage pools residing on an IBM Spectrum Protect server.

Both functions require an open path for communication between the nodes designated for use with **mmbackup** or HSM policies and the external IBM Spectrum Protect server. The port requirement listed in the following table can be viewed in the `dsm.sys` configuration file also.

Table 75. Required port number for mmbackup and HSM connectivity to IBM Spectrum Protect server

Port number	Protocol	Service name	Components involved in communication
1500	TCP	TSM	IBM Spectrum Protect Backup-Archive client communication with server

For information on port requirements specific to the server end, see IBM Spectrum Protect documentation on the IBM Knowledge Center.

Firewall considerations for using IBM Spectrum Archive with IBM Spectrum Scale

The IBM Spectrum Archive software is installed on a node or a group of nodes in an IBM Spectrum Scale cluster.

This requires that each IBM Spectrum Archive node can communicate with the rest of the cluster using the ports required for basic GPFS cluster operations. For more information, see "Firewall recommendations for internal communication among nodes" on page 689. In addition to this, IBM Spectrum Archive communicates with RPC. For RPC related port requirements, see "Firewall recommendations for protocol access" on page 690.

IBM Spectrum Archive can connect to tape drives using a SAN or a direct connection.

Firewall recommendations for file audit logging

This topic describes port access and firewall protection during file audit logging activities.

The file audit logs use the following ports.

Table 76. Recommended port numbers that can be used for file audit logging

Port Number	Protocol	Service Name	Components involved in communication
9092	TCP	IBM Spectrum Scale	File audit logging
9093	TCP	IBM Spectrum Scale	File audit logging
2181	TCP	IBM Spectrum Scale	File audit logging
2888 - 3888 (1000 ports)	TCP	IBM Spectrum Scale	File audit logging

Examples of how to open firewall ports

Use these examples as a reference for opening firewall ports on different operating systems, if required. It is recommended to restrict port traffic to only the required network or adapters.

Red Hat Enterprise Linux 7.x and CentOS 7.x

- Issue the following command to list currently open ports.

  `firewall-cmd --list-ports`

- Issue the following command to list zones.

  `firewall-cmd --get-zones`

- Issue the following command to list the zone containing `eth0`.

  `firewall-cmd --get-zone-of-interface=eth0`

- Issue the following command to open port 1191 for TCP traffic.

  `firewall-cmd --add-port 1191/tcp`

- Issue the following command to open port 1191 for TCP traffic after reboot. Use this command to make changes persistent.

  `firewall-cmd --permanent --add-port 1191/tcp`

- Issue the following command to open a range a range of ports.

  `firewall-cmd --permanent --add-port 60000-61000/tcp`

- Issue the following command to stop and start the firewall.

  `systemctl stop firewalld`

  `systemctl start firewalld`

SLES 12

1. Open the YaST tool by issuing the following command: `yast`
2. Click **Security and Users** > **Firewall**.
3. Select the **Allowed Services** tab and click **Advanced...**.
4. Enter the desired port range in the `from-port-start:to-port-end` format and specify the protocol (TCP or UDP). For example, enter `60000:60010` to open ports 60000 to 60010.
5. Click **OK** to close the Advanced dialog box.
6. Click **Next** and review the summary of your changes.
7. Click **Finish** to apply your changes.

Ubuntu and Debian

- Issue the following command to open port 1191 for TCP traffic.

  `sudo ufw allow 1191/tcp`

- Issue the following command to open a range of ports.

  `sudo ufw allow 60000:61000/tcp`

- Issue the following command to stop and start Uncomplicated Firewall (UFW).

  `sudo ufw disable`

  `sudo ufw enable`

Microsoft Windows 2008 R2

1. Open the Windows Firewall utility: **Control Panel** > **Administrative Tools** > **Windows Firewall with Advanced Security**
2. Add new inbound and outbound rules as required.

Firewall configuration using iptables

The `iptables` utility is available on most Linux distributions to set firewall rules and policies. These Linux distributions include Red Hat Enterprise Linux 6.8, Red Hat Enterprise Linux 7.x, CentOS 7.x, SLES 12, Ubuntu, and Debian. Before using these commands, check which firewall zones might be enabled by

default. Depending upon the zone setup, the INPUT and OUTPUT terms might need to be renamed to match a zone for the desired rule. See the following Red Hat Enterprise Linux 7.x example for one such case.

- Issue the following command to list the current firewall policies.

  ```
  sudo iptables -S
  sudo iptables -L
  ```

- Issue the following command to open port 1191 (GPFS) for inbound TCP traffic from internal subnet 172.31.1.0/24.

  ```
  sudo iptables -A INPUT -p tcp -s 172.31.1.0/24 --dport 1191 -j ACCEPT
  ```

- Issue the following command to open port 1191 (GPFS) for outbound TCP traffic to internal subnet 172.31.1.0/24.

  ```
  sudo iptables -A OUTPUT -p tcp -d 172.31.1.0/24 --sport 1191 -j ACCEPT
  ```

- Issue the following command to open port 445 (SMB) for outbound TCP traffic to external subnet 10.11.1.0/24 and only for adapter eth1.

  ```
  sudo iptables -A OUTPUT -o eth1 -p tcp -d 10.11.1.0/24 --sport 445 -j ACCEPT
  ```

- Issue the following command to open port 445 (SMB) for inbound TCP traffic to a range of CES IPs (10.11.1.5 through 10.11.1.11) and only for adapter eth1.

  ```
  sudo iptables -A INPUT -i eth1 -p tcp -m iprange --dst-range 10.11.1.5-10.11.1.11 --dport 445 -j ACCEPT
  ```

- Issue the following command to allow an internal network, eth1, to communicate with an external network, eth0.

  ```
  sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
  ```

- [Red Hat Enterprise Linux 7.x specific] Issue the following command to open Chef port 8889 for inbound traffic from subnet 10.18.0.0/24 on eth1 within the public zone.

  ```
  iptables -A IN_public_allow -i eth1 -p tcp -s 10.18.0.0/24 --dport 8889 -j ACCEPT
  ```

- Issue the following command to save firewall rule changes to persist across a reboot.

  ```
  sudo iptables-save
  ```

- Issue the following command to stop and start Uncomplicated Firewall (UFW).

  ```
  service iptables stop
  service iptables start
  ```

For information on how CES IPs are aliased to network adapters, see "CES IP aliasing to network adapters on protocol nodes" on page 27.

Logging file system activities

File system activities such as file creation, deletion, and renaming might need to be logged. Varonis DatAdvantage software can be used to log file activity within IBM Spectrum Scale.

Special administrative exports are required for each file system the user needs to detect and log file system activity for. This also necessitates the use of Active Directory as the file authentication method for the IBM Spectrum Scale cluster.

Ensure that the prerequisites are met before configuring the IBM Spectrum Scale node and installing the Varonis agent. Once the agent is successfully installed, install the DatAdvantage software and the management console. Tune Varonis DatAdvantage to work with IBM Spectrum Scale to audit the file system activity. For more information on Varonis DatAdvantage, and how it can be incorporated with IBM Spectrum Scale, see Varonis Audit Logging.

Note:
To view the Varonis Audit Logging information in the IBM developer works website, non-IBM users have to register and create a username and password to access the information. IBM users must scroll to the bottom of the page and use the link to login with their IBM intranet username and password.

Supported web browser versions and web browser settings for GUI

To access the management GUI, you must ensure that your web browser is supported and has the appropriate settings enabled.

The management GUI supports the following web browsers:
- Mozilla Firefox 5152
- Mozilla Firefox Extended Support Release (ESR) 4552
- Microsoft Internet Explorer (IE) 11
- Google Chrome 59

IBM supports higher versions of the browsers if the vendors do not remove or disable function that the product relies upon. For browser levels higher than the versions that are certified with the product, customer support accepts usage-related and defect-related service requests. If the support center cannot re-create the issue, support might request the client to re-create the problem on a certified browser version. Defects are not accepted for cosmetic differences between browsers or browser versions that do not affect the functional behavior of the product. If a problem is identified in the product, defects are accepted. If a problem is identified with the browser, IBM might investigate potential solutions or work-arounds that the client can implement until a permanent solution becomes available.

To configure your web browser, follow these steps:
1. Enable JavaScript for your web browser.

   For Mozilla Firefox, JavaScript is enabled by default and requires no additional configuration.

   For Microsoft Internet Explorer (IE) running on Microsoft Windows 7:
   a. In Internet Explorer, click **Tools** > **Internet Options**.
   b. Click **Security Settings**.
   c. Click **Internet** to choose the Internet zone.
   d. Click **Custom Level**.
   e. Scroll down to the **Scripting** section, and then in **Active Scripting**, click **Enable**.
   f. Click **OK** to close **Security Settings**.
   g. Click **Yes** to confirm the change for the zone.
   h. Click **OK** to close **Internet Options**.
   i. Refresh your browser.

   For Microsoft Internet Explorer (IE) running on Microsoft Windows Server 2008:
   a. In Internet Explorer, click **Tools** > **Internet Options**.
   b. Click **Security**.
   c. Click **Trusted sites**.
   d. On the **Trusted sites** dialog, verify that the web address for the management GUI is correct and click **Add**.
   e. Verify that the correct web address was added to the **Trusted sites** dialog.
   f. Click **Close** on the **Trusted sites** dialog.
   g. Click **OK**.
   h. Refresh your browser.

   For Google Chrome:
   a. On the menu bar in the Google Chrome browser window, click **Settings**.
   b. Click **Show advanced settings**.
   c. In the **Privacy** section, click **Content settings**.
   d. In the **JavaScript** section, select **Allow all sites to run JavaScript**.

e. Click **OK**.

f. Refresh your browser.

2. Enable cookies in your web browser.

   For Mozilla Firefox:

   a. On the menu bar in the Firefox browser window, click **Tools** > **Options**.

   b. On the Options window, select **Privacy**.

   c. Set "Firefox will" to **Use custom settings for history**.

   d. Select **Accept cookies from sites** to enable cookies.

   e. Click **OK**.

   f. Refresh the browser.

   For Microsoft Internet Explorer:

   a. In Internet Explorer, click **Tools** > **Internet Options**.

   b. Click **Privacy**. Under **Settings**, move the slider to the bottom to allow all cookies.

   c. Click **OK**.

   d. Refresh your browser.

   For Google Chrome:

   a. On the menu bar in the Google Chrome browser window, click **Settings**.

   b. Click **Show advanced settings**.

   c. In the **Privacy** section, click **Content settings**.

   d. In the **Cookies** section, select **Allow local data to be set**.

   e. Click **OK**.

   f. Refresh your browser.

3. Enable file download on IE that runs on Windows 2012.

   a. In Internet Explorer, click **Tools** > **Internet Options**.

   b. On the Internet Options window, select the **Security** tab.

   c. On the **Security** tab, click the **Internet zone**.

   d. Click **Custom level** to customize the security level for this zone.

   e. Scroll down to **Downloads** and select **Enable** under File download.

   f. Click **OK**.

   g. Click **Yes** to confirm.

   h. Click **OK** to close the Internet Options.

4. Enable scripts to disable or replace context menus. (Mozilla Firefox only).

   For Mozilla Firefox:

   a. On the menu bar in the Firefox browser window, click **Tools** > **Options**.

   b. On the Options window, select **Content**.

   c. Click **Advanced** by the **Enable JavaScript** setting.

   d. Select **Disable or replace context menus**.

   e. Click **OK** to close the Advanced window.

   f. Click **OK** to close the Options window.

   g. Refresh your browser.

Chapter 43. GUI limitations

The following are the limitations of the IBM Spectrum Scale GUI:

1. The GUI supports RHEL 7.x, SLES 12.x, and Ubuntu 16.x on Intel x86, Power (Big or Little Endian), and IBM Z platforms.
2. Up to 1000 nodes are supported.
3. The GUI supports a subset of the CLI functionality. Additional capabilities will be added in the future releases of the product.
4. The Object management panels do not support configurations with Keystone V2 API.
5. One GUI instance supports a single cluster.
6. In an IBM Spectrum Scale and Elastic Storage Server (ESS) mixed support environment, the ESS GUI must manage the whole cluster to display the ESS-specific pages in the GUI.
7. The GUI is not supported on the cluster that runs on IBM Spectrum Scale release earlier than 4.2.0.0. The GUI supports IBM Spectrum Scale release 4.2.0.0 or later. Issue the `mmlsconfig` command to see the value that is set for the `minReleaseLevel` attribute. Use the `mmchconfig release=LATEST` command and restart the GUI to make the management GUI fully operational at the new code level. As changing the minimum release level affects the cluster behavior, refer the `mmchconfig` command man page and other related topics before you make this configuration change.
8. The GUI node must be a homogeneous stack. That is, all packages must be of the same release. For example, do not mix the 5.0.0 GUI rpm with a 4.2.3 base rpm. However, GUI PTFs and efixes can usually be applied without having to install the corresponding PTF or efix of the base package. This is helpful if you just want to get rid of a GUI issue without changing anything on the base layer.

For limitations of the installation GUI, see *Installing IBM Spectrum Scale by using the graphical user interface (GUI)* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Accessibility features for IBM Spectrum Scale

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 Spectrum Scale:
- 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 Knowledge Center, and its related publications, are accessibility-enabled. The accessibility features are described in IBM Knowledge Center (www.ibm.com/support/knowledgecenter).

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 US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (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

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide 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 Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of 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.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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 actual people or business enterprises 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.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).
Portions of this code are derived from IBM Corp.
Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

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.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not 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 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 may not download, export or re-export 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.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to

collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Glossary

This glossary provides terms and definitions for IBM Spectrum Scale.

The following cross-references are used in this glossary:

- *See* refers you from a nonpreferred 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 (www.ibm.com/software/globalization/terminology) (opens in new window).

B

block utilization
    The measurement of the percentage of used subblocks per allocated blocks.

C

cluster
    A loosely-coupled collection of independent systems (nodes) organized into a network for the purpose of sharing resources and communicating with each other. See also *GPFS cluster*.

cluster configuration data
    The configuration data that is stored on the cluster configuration servers.

Cluster Export Services (CES) nodes
    A subset of nodes configured within a cluster to provide a solution for exporting GPFS file systems by using the Network File System (NFS), Server Message Block (SMB), and Object protocols.

cluster manager
    The node that monitors node status using disk leases, detects failures, drives recovery, and selects file system managers. The cluster manager must be a quorum node. The selection of the cluster manager node favors the quorum-manager node with the lowest node number among the nodes that are operating at that particular time.

> **Note:** The cluster manager role is not moved to another node when a node with a lower node number becomes active.

control data structures
    Data structures needed to manage file data and metadata cached in memory. Control data structures include hash tables and link pointers for finding cached data; lock states and tokens to implement distributed locking; and various flags and sequence numbers to keep track of updates to the cached data.

D

Data Management Application Program Interface (DMAPI)
    The interface defined by the Open Group's XDSM standard as described in the publication *System Management: Data Storage Management (XDSM) API Common Application Environment (CAE) Specification C429*, The Open Group ISBN 1-85912-190-X.

deadman switch timer
    A kernel timer that works on a node that has lost its disk lease and has outstanding I/O requests. This timer ensures that the node cannot complete the outstanding I/O requests (which would risk causing file system corruption), by causing a panic in the kernel.

dependent fileset
    A fileset that shares the inode space of an existing independent fileset.

disk descriptor
    A definition of the type of data that the disk contains and the failure group to which this disk belongs. See also *failure group*.

disk leasing
    A method for controlling access to storage devices from multiple host systems. Any host that wants to access a storage device configured to use disk leasing registers for a lease; in the event of a perceived failure, a host system can deny access,

preventing I/O operations with the storage device until the preempted system has reregistered.

disposition

The session to which a data management event is delivered. An individual disposition is set for each type of event from each file system.

domain

A logical grouping of resources in a network for the purpose of common management and administration.

E

ECKD™

See *extended count key data (ECKD)*.

ECKD device

See *extended count key data device (ECKD device)*.

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*, *master encryption key*.

extended count key data (ECKD)

An extension of the count-key-data (CKD) architecture. It includes additional commands that can be used to improve performance.

extended count key data device (ECKD device)

A disk storage device that has a data transfer rate faster than some processors can utilize and that is connected to the processor through use of a speed matching buffer. A specialized channel program is needed to communicate with such a device. See also *fixed-block architecture disk device*.

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*.

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.

file clone

A writable snapshot of an individual file.

file encryption key (FEK)

A key used to encrypt sectors of an individual file. See also *encryption key*.

file-management policy

A set of rules defined in a policy file that GPFS uses to manage file migration and file deletion. See also *policy*.

file-placement policy

A set of rules defined in a policy file that GPFS uses to manage the initial placement of a newly created file. See also *policy*.

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.

fixed-block architecture disk device (FBA disk device)
A disk device that stores data in blocks of fixed size. These blocks are addressed by block number relative to the beginning of the file. See also *extended count key data device*.

fragment
The space allocated for an amount of data too small to require a full block. A fragment consists of one or more subblocks.

G

global snapshot
A snapshot of an entire GPFS file system.

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 recovery log
A file that contains a record of metadata activity, and exists for each node of a cluster. In the event of a node failure, the recovery log for the failed node is replayed, restoring the file system to a consistent state and allowing other nodes to continue working.

I

ill-placed file
A file assigned to one storage pool, but having some or all of its data in a different storage pool.

ill-replicated file
A file with contents that are not correctly replicated according to the desired setting for that file. This situation occurs in the interval between a change in the file's replication settings or suspending one of its disks, and the restripe of the file.

independent fileset
A fileset that has its own inode space.

indirect block
A block containing 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.

ISKLM
IBM Security Key Lifecycle Manager. For GPFS encryption, the ISKLM is used as an RKM server to store MEKs.

J

journaled file system (JFS)
A technology designed for high-throughput server environments, which are important for running intranet and other high-performance e-business file servers.

junction
A special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

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.

M

master encryption key (MEK)
A key used to encrypt other keys. See also *encryption key*.

MEK See *master encryption key*.

metadata
Data structures that contain information that is needed to access file data. Metadata includes inodes, indirect blocks, and directories. Metadata is not accessible to user applications.

metanode
The one node per open file that is responsible for maintaining file metadata integrity. In most cases, the node that has had the file open for the longest period of continuous time is the metanode.

mirroring
The process of writing the same data to multiple disks at the same time. The

mirroring of data protects it against data loss within the database or within the recovery log.

Microsoft Management Console (MMC)

A Windows tool that can be used to do basic configuration tasks on an SMB server. These tasks include administrative tasks such as listing or closing the connected users and open files, and creating and manipulating SMB shares.

multi-tailed

A disk connected to multiple nodes.

N

namespace

Space reserved by a file system to contain the names of its objects.

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 hex 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 may contain one or more nodes.

node descriptor

A definition that indicates how GPFS uses a node. Possible functions include: manager node, client node, quorum node, and nonquorum node.

node number

A number that is generated and maintained by GPFS 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 GPFS 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.

P

policy A list of file-placement, service-class, and encryption rules that define characteristics and placement of files. Several policies can be defined within the configuration, but only one policy set is active at one time.

policy rule

A programming statement within a policy that defines a specific action to be performed.

pool A group of resources with similar characteristics and attributes.

portability

The ability of a programming language to compile successfully on different operating systems without requiring changes to the source code.

primary GPFS cluster configuration server

In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data.

private IP address

A IP address used to communicate on a private network.

public IP address

A IP address used to communicate on a public network.

Q

quorum node

A node in the cluster that is counted to determine whether a quorum exists.

quota The amount of disk space and number of inodes assigned as upper limits for a specified user, group of users, or fileset.

quota management

The allocation of disk blocks to the other nodes writing to the file system, and comparison of the allocated space to quota limits at regular intervals.

R

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.

remote key management server (RKM server)
A server that is used to store master encryption keys.

replication
The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target), and synchronizing the data in both locations.

RKM server
See *remote key management server*.

rule A list of conditions and actions that are triggered when certain conditions are met. Conditions include attributes about an object (file name, type or extension, dates, owner, and groups), the requesting client, and the container name associated with the object.

S

SAN-attached
Disks that are physically attached to all nodes in the cluster using Serial Storage Architecture (SSA) connections or using Fibre Channel switches.

Scale Out Backup and Restore (SOBAR)
A specialized mechanism for data protection against disaster only for GPFS file systems that are managed by IBM Spectrum Protect Hierarchical Storage Management (HSM).

secondary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data in the event that the primary GPFS cluster configuration server fails or becomes unavailable.

Secure Hash Algorithm digest (SHA digest)
A character string used to identify a GPFS security key.

session failure
The loss of all resources of a data management session due to the failure of the daemon on the session node.

session node
The node on which a data management session was created.

Small Computer System Interface (SCSI)
An ANSI-standard electronic interface that allows personal computers to communicate with peripheral hardware, such as disk drives, tape drives, CD-ROM drives, printers, and scanners faster and more flexibly than previous interfaces.

snapshot
An exact copy of changed data in the active files and directories of a file system or fileset at a single point in time. See also *fileset snapshot*, *global snapshot*.

source node
The node on which a data management event is generated.

stand-alone client
The node in a one-node cluster.

storage area network (SAN)
A dedicated storage network tailored to a specific environment, combining servers, storage products, networking products, software, and services.

storage pool
A grouping of storage space consisting of volumes, logical unit numbers (LUNs), or addresses that share a common set of administrative characteristics.

stripe group
The set of disks comprising the storage assigned to a file system.

striping
A storage process in which information is split into blocks (a fixed amount of data) and the blocks are written to (or read from) a series of disks in parallel.

subblock
>The smallest unit of data accessible in an I/O operation, equal to one thirty-second of a data block.

system storage pool
>A storage pool containing file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks and extended attributes The **system storage pool** can also contain user data.

T

token management
>A system for controlling file access in which each application performing a read or write operation is granted some form of access to a specific block of file data. Token management provides data consistency and controls conflicts. Token management has two components: the token management server, and the token management function.

token management function
>A component of token management that requests tokens from the token management server. The token management function is located on each cluster node.

token management server
>A component of token management that controls tokens relating to the operation of the file system. The token management server is located at the file system manager node.

transparent cloud tiering (TCT)
>A separately installable add-on feature of IBM Spectrum Scale that provides a native cloud storage tier. It allows data center administrators to free up on-premise storage capacity, by moving out cooler data to the cloud storage, thereby reducing capital and operational expenditures. .

twin-tailed
>A disk connected to two nodes.

U

user storage pool
>A storage pool containing the blocks of data that make up user files.

V

VFS See *virtual file system*.

virtual file system (VFS)
>A remote file system that has been mounted so that it is accessible to the local user.

virtual node (vnode)
>The structure that contains information about a file system object in a virtual file system (VFS).

Index

Special characters

A

IBM ®