Reliable Scalable Cluster Technology
Version 3.1.5.0

Administering RSCT

Reliable Scalable Cluster Technology
Version 3.1.5.0

Administering RSCT

Note
Before using this information and the product it supports, read the information in "Notices" on page 319.

This edition applies to Reliable Scalable Cluster Technology Version 3.1.5.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this document

IBM® Reliable Scalable Cluster Technology (RSCT) is a set of software components that together provide a comprehensive clustering environment for IBM AIX®, Linux, Solaris, and Windows nodes. This information describes various component subsystems of RSCT:

- The resource monitoring and control (RMC) subsystem and core resource managers that together enable you to monitor various resources of your system and create automated responses to changing conditions of those resources.
- How to use the configuration resource manager to configure a set of nodes into a cluster for high availability. Such a cluster is called an RSCT peer domain.
- How to use the least-privilege resource manager to control access to root commands and scripts by allowing specific non-root users to be authorized to run specific commands requiring root authority.
- How to use the storage resource manager to monitor and control the storage resources within an RSCT peer domain.
- The basics of cluster security services, which are used by other RSCT components and other cluster products for authentication, and common administration tasks associated with the cluster security services.
- The Topology Services subsystem, which provides other subsystems with network adapter status, node connectivity information, and a reliable messaging service.
- The Group Services subsystem, which provides other component subsystems with a distributed coordination and synchronization service.

Highlighting

The following highlighting conventions are used in this document:

Table 1. Conventions

Convention	Usage
bold	**Bold** words or characters represent system elements that you must use literally, such as commands, flags, path names, directories, file names, values, PE component names (**poe**, for example), and selected menu options.
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.
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 d "FileSystem space used"`

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

Table 1. Conventions (continued)

Convention	Usage
|	• In *synopsis* or *syntax* 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.

Entering commands

When you work with the operating system, you typically enter commands following the shell prompt on the command line. The shell prompt can vary. In the following examples, $ is the prompt.

To display a list of the contents of your current directory, you would type `ls` and press the **Enter** key:

```
$ ls
```

When you enter a command and it is running, the operating system does not display the shell prompt. When the command completes its action, the system displays the prompt again. This indicates that you can enter another command.

The general format for entering operating system commands is:

Command Flag(s) Parameter

The flag alters the way a command works. Many commands have several flags. For example, if you type the `-l` (long) flag following the **ls** command, the system provides additional information about the contents of the current directory. The following example shows how to use the `-l` flag with the **ls** command:

```
$ ls -l
```

A parameter consists of a string of characters that follows a command or a flag. It specifies data, such as the name of a file or directory, or values. In the following example, the directory named `/usr/bin` is a parameter:

```
$ ls -l /usr/bin
```

When entering commands in, it is important to remember the following items:

- Commands are usually entered in lowercase.
- Flags are usually prefixed with a - (minus sign).
- More than one command can be typed on the command line if the commands are separated by a ; (semicolon).
- Long sequences of commands can be continued on the next line by using the \ (backslash). The backslash is placed at the end of the first line. The following example shows the placement of the backslash:

  ```
  $ cat /usr/ust/mydir/mydata > \
  /usr/usts/yourdir/yourdata
  ```

When certain commands are entered, the shell prompt changes. Because some commands are actually programs (such as the telnet command), the prompt changes when you are operating within the command. Any command that you issue within a program is known as a subcommand. When you exit the program, the prompt returns to your shell prompt.

The operating system can operate with different shells (for example, Bourne, C, or Korn) and the commands that you enter are interpreted by the shell. Therefore, you must know what shell you are using so that you can enter the commands in the correct format.

Case sensitivity in AIX

Everything in the AIX operating system is case sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type `LS`, the system responds that the command `is not found`. Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

RSCT versions

This edition applies to RSCT version, release, modification, and fix number 3.1.5.0.

You can use the **ctversion** command to find out which version of RSCT is running on a particular AIX, Linux, Solaris, or Windows node. For example:

```
/usr/sbin/rsct/install/bin/ctversion
```

An example of the output follows:

```
# /usr/sbin/rsct/install/bin/ctversion
rlis1313a 3.1.5.0
```

where, `rlis1313a` is the RSCT build level.

On the AIX operating system, you can also use the **lslpp** command to find out which version of RSCT is running on a particular AIX node. For example:

```
lslpp -L rsct.core.utils
```

An example of the output follows:

```
  Fileset                      Level    State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  rsct.core.utils              3.1.5.0    C     F    RSCT Utilities

State codes:
 A -- Applied.
 B -- Broken.
 C -- Committed.
 E -- EFIX Locked.
 O -- Obsolete.  (partially migrated to newer version)
 ? -- Inconsistent State...Run lppchk -v.

Type codes:
 F -- Installp Fileset
 P -- Product
 C -- Component
 T -- Feature
 R -- RPM Package
```

On the Linux operating system, you can also use the **rpm** command to find out which version of RSCT is running on a particular Linux or Solaris node. For example:

```
rpm -qa | grep rsct.basic
```

On the Windows operating system, you can also perform the following steps to find out which version of RSCT is running on a particular Windows node:

1. Click the Windows **start** button.

2. Select **All Programs**.
3. Select **Tivoli SA MP Base**.
4. Click **SA MP Version**.

Related information

The following PDF documents that contain RSCT information can be found at Reliable Scalable Cluster Technology (RSCT) PDFs:

- *Messages for RSCT*
- *Programming Group Services for RSCT*
- *Programming RMC for RSCT*
- *Technical Reference: RSCT for AIX*
- *Technical Reference: RSCT for Multiplatforms*
- *Troubleshooting RSCT*

Administering RSCT

This information should be read by anyone who wants to:

- Understand the core RSCT components
- Configure a set of nodes into an RSCT peer domain
- Understand how authentication is handled by cluster security services and administer cluster security
- Understand and diagnose problems with Topology Services
- Understand and diagnose problems with Group Services

What's new in Administering RSCT

Read about new or significantly changed information for the Administering RSCT topic collection.

How to see what's new or changed

In this PDF file, you might see revision bars (|) in the left margin that identifies new and changed information.

June 2014

The following information is a summary of the updates made to this topic collection:

- Updated the **lsrsrc** command usage in the "Defining preferred nodes for Group Services name server and group leader selection" on page 83 topic.
- Obsolete information was removed or changed in various topics.

November 2013

The following information is a summary of the updates made to this topic collection:

- Updated the libraries and packages for the supported version of Linux distributions in the "Verifying RSCT installation on Linux nodes" on page 27 topic.
- Updated the supported versions of Linux distributions in the "Supported Linux distributions for RSCT" on page 31 topic.
- Added information about the SRC 64-bit library package in the "Verifying RSCT installation on Linux nodes" on page 27 and "Verifying RSCT installation on Solaris nodes" on page 32 topics.
- Updated policy information in the "Creating a peer domain definition" on page 52 and "Examples for creating a peer domain definition" on page 55 topics.
- Added an example to the "Adding nodes to the peer domain" on page 62 topic.
- Updated the "Adding nodes to a peer domain in a CAA environment" on page 64 topic.
- Updated an example in the "Setting the critical resource protection method for a peer domain or a node in a peer domain" on page 86 topic.
- Added a note in the "Target nodes for a command" on page 125 topic stating that RMC commands and client applications that use local or management domain scope do not lose their connection to RMC when the node is brought online or taken offline.
- Updated information about the following attributes in the "How the storage resource manager monitors the availability of shared storage" on page 243 topic:
  - **IBM.Disk DeviceLockMode** attribute
  - **IBM.VolumeGroup DeviceLockMode** attribute
  - **IBM.VolumeGroup AutoMonitor** attribute

  - **IBM.AgFileSystem VGStateInfo** attribute
- Added the "Trace summary" on page 249 topic.

March 2013

The following information is a summary of the updates made to this topic collection:

- Updated information about the **IBM.MgmtDomainRM** resource class. For more information, see "Management domain configuration" on page 21.

November 2012

The following information is a summary of the updates made to this topic collection:

- Added a new section about RSCT peer domain support on Cluster Aware AIX (CAA) linked clusters. Information was also added about new persistent attributes for the **IBM.PeerNode** resource class: **CriticalMode** and **NotifyQuorumChangedCommand**. For more information, see "Peer domain on CAA linked clusters" on page 41.
- Added information about the **CriticalMode** persistent attribute in "Setting the critical resource protection method for a peer domain or a node in a peer domain" on page 86. It is introduced as an enhancement to critical protection in a peer domain partition.
- Added the AIX 6.1 platform support information for the tiebreaker type small computer system interface persistent reservation (SCSIPR) throughout the topic collection. SCSIPR, which was specific to Linux on the System x® hardware platform, is now supported on the AIX 6.1 platform, and later.
- Added information about the **QuorumLess** quorum type mode in the following topics:
  - "How the configuration resource manager uses a tiebreaker to determine operational quorum" on page 39
  - "Resource classes defined by the configuration resource manager" on page 114
- Updated the quorum rule for the **QuorumLess** quorum type in "Quorum types" on page 39.
- Added information about the third predefined tiebreaker, **Success**, in the following topics:
  - "Setting the active tiebreaker" on page 89
  - "Defining a new tiebreaker" on page 91
- Added the **-t** flag to the **tb_break** command usage for all of the tiebreaker types throughout the topic collection.
- Added a new section about monitoring the critical file system and **crit_fs.conf** file. For more information, see "Monitoring critical file system" on page 105.

October 2011

The following information is a summary of the updates made to this topic collection:

- Enhanced support for Cluster-Aware AIX (CAA).
  - Better integration with shared storage pool (SSP) and PowerHA® environments.
  - Internal enhancements for stability and performance.

  For more information, see "Quorumless domain" on page 37.
- Support for **QuorumLess** quorum type as a class attribute for the **IBM.PeerDomain** resource class and as a persistent attribute for the **IBM.PeerNode** resource class for the specific environments (PowerHA and SSP). For more information, see "Quorum types" on page 39.
- New persistent attribute for the **IBM.AgFileSystem** resource class: **PreOfflineMethod**. For more information, see "Configuring method used to resolve problems when initial file system mount fails" on page 255.

- Changes to the configuration resource manager (ConfigRM) component to exclude the link-local IPv4 and IPv6 addresses from heartbeating and the trusted host list (THL) files. For more information, see "Changing IP addresses in a peer domain" on page 68.
- New tiebreaker type: small computer system interface persistent reservation (SCSIPR), which is specific to Linux on the System x hardware platform, is introduced by IBM Tivoli® System Automation for MultiPlatforms 3.2.1.2. For more information, see "Creating an SCSIPR tiebreaker" on page 99.

RSCT concepts

Reliable Scalable Cluster Technology (RSCT) is a set of software components that together provide a comprehensive clustering environment for AIX, Linux, Solaris, and Windows operating systems. RSCT is the infrastructure used by a variety of IBM products to provide clusters with improved system availability, scalability, and ease of use.

RSCT includes the following components:

- **Resource monitoring and control (RMC) subsystem**. This is the scalable, reliable backbone of RSCT. It runs on a single machine or on each node (operating system image) of a cluster and provides a common abstraction for the resources of the individual system or the cluster of nodes. You can use RMC for single system monitoring or for monitoring nodes in a cluster. In a cluster, however, RMC provides global access to subsystems and resources throughout the cluster, thus providing a single monitoring and management infrastructure for clusters.
- **RSCT core resource managers**. A resource manager is a software layer between a resource (a hardware or software entity that provides services to some other component) and RMC. A resource manager maps programmatic abstractions in RMC into the actual calls and commands of a resource.
- **RSCT cluster security services**. This RSCT component provides the security infrastructure that enables RSCT components to authenticate the identity of other parties.
- **Topology Services subsystem**. This RSCT component provides node and network failure detection on some cluster configurations.
- **Group Services subsystem**. This RSCT component provides cross-node/process coordination on some cluster configurations.

Management domains and peer domains

The set of nodes that is configured for manageability or monitoring is called a *management domain* of your cluster. The set of nodes that is configured for high availability is called an RSCT peer domain of your cluster.

A *peer domain* is a set of nodes that have a consistent knowledge of the existence of each other and of the resources shared among them. On each node within the peer domain, RMC depends on a core set of cluster services, which include Topology Services, Group Services and cluster security services.

A *management domain* is a set of nodes with resources that can be managed and monitored from one of the nodes, which is designated as the management control point (MCP). All other nodes are considered to be managed nodes. Topology Services and Group Services are not used in a management domain.

In order to understand how the various RSCT components are used in a cluster, be aware that nodes of a cluster can be configured for manageability, high availability or both.

Configure a set of nodes for manageability or monitoring using the Cluster Systems Management (CSM) licensed program, as described in the *Cluster Systems Management: Administration Guide*, or using the Extreme Cloud Administration Tookit (xCAT), as described in the xCAT documentation found at **http://xcat.sourceforge.net**. You can also create a management domain manually using the **mkrsrc** command. Note that logical partitions (LPARs) on Power Systems™ servers are automatically configured as managed nodes in a management domain in which an MCP is the management console.

Configure a set of nodes for high availability using RSCT's configuration resource manager.

Table 2 lists the characteristics of the two domain types — management domains and peer domains — that can be present in your cluster.

Table 2. Characteristics of management domains and peer domains

Management domain characteristics	Peer domain characteristics
Typically established and administered by CSM or xCAT or can be created manually using the **mkrsrc** command. Logical partitions (LPARs) on Power Systems servers are automatically configured as managed nodes to the management consoles.	Established and administered by the RSCT configuration resource manager.
Has a management server that is used to administer a number of managed nodes. Only management servers have knowledge of the whole domain. Managed nodes only know about the servers managing them. Managed nodes know nothing of each other. A management server is also known as a management control point (MCP).	Consists of a number of nodes with no distinguished or master node. All nodes are aware of all other nodes and administration commands can be issued from any node in the domain. All nodes have a consistent view of the domain membership.
A managed node can exist simultaneously in multiple management domains, responding to commands from each MCP.	A peer node can be defined to multiple peer domains, but can be active (online) in only one peer domain.
Processor architecture and operating system are heterogeneous.	Processor architecture and operating system are heterogeneous. Some programs that are designed to run in a peer domain might not support the same heterogeneous environment as RSCT. See the specific exploiter's documentation for information on supported processor architecture and operating systems.
RSCT cluster security services are used to authenticate other parties.	RSCT cluster security services are used to authenticate other parties.
The Topology Services subsystem is *not* needed.	The Topology Services subsystem provides node/network failure detection.
The Group Services subsystem is *not* needed.	The Group Services subsystem provides cross node/process coordination.

Although your cluster may be divided into management and peer domains, keep in mind that an individual node can participate in both domain types. However, within a management domain, a management server cannot belong to the same peer domain as any of the managed nodes.

Related concepts:

"Target nodes for a command" on page 125
Resource monitoring and control (RMC) is a daemon that runs on individual systems or on each node of a cluster. It provides a single management and monitoring infrastructure for individual nodes, peer domains, and management domains.

The resource monitoring and control subsystem

The resource monitoring and control (RMC) subsystem is the scalable backbone of RSCT that provides a generalized framework for managing resources within a single system or a cluster.

Its generalized framework is used by cluster management tools to monitor, query, modify, and control cluster resources. RMC provides a single monitoring and management infrastructure for both RSCT peer domains and management domains. RMC can also be used on a single machine, enabling you to monitor and manage the resources of that machine. However, when a group of machines, each running RMC, are clustered together (into management domains or peer domains), the RMC framework allows a process on any node to perform an operation on one or more *resources* on any other node in the domain. A *resource* is the fundamental concept of the RMC architecture; it is an instance of a physical or logical entity that provides services to some other component of the system. Examples of resources include **lv01** on node 10,

Ethernet device **en0** on node 14, and IP address 9.117.7.21. A set of resources that have similar characteristics (in terms of services provided, configuration parameters, and so on) is called a *resource class*.

The resources and resource class abstractions are defined by a *resource manager*.

RSCT resource managers

A *resource manager* is a process that maps resource and resource class abstractions into actual calls and commands for one or more specific types of resources. A resource manager runs as a stand-alone daemon and contains definitions of all resource classes that the resource manager supports. These definitions include a descriptions of all attributes, actions, and other characteristics of a resource class.

An RMC access control list (ACL) defines the access permissions that authorized users have for manipulating and grouping a resource class. For more information on RMC, see "Monitoring resources using RMC and resource managers" on page 126.

RSCT provides a core set of resource managers for managing base resources on single systems and across clusters. Additional resource managers are provided by cluster licensed programs.

Some resource managers provide lower-level instrumentation and control of system resources. Others are essentially management applications implemented as resource managers.

The core RSCT resource managers are:

- **Audit log resource manager**, which provides a system-wide facility for recording information about the system's operation. This is particularly useful for tracking subsystems running in the background. A command-line interface to the resource manager enables you to list and remove records from an audit log.
- **CIM resource manager**, which enables you to use RMC to query system configuration through Common Information Model (CIM) classes. For more information, see "Querying and monitoring CIM properties and associations" on page 184.
- **Configuration resource manager**, which provides the ability to create and administer an RSCT peer domain. This is essentially a management application implemented as a resource manager. A command-line interface to this resource manager enables you to create a new peer domain, add nodes to the domain, list nodes in the domain, and so on. See "Administering an RSCT peer domain" on page 35 for more information.
- **Event response resource manager**, which provides the ability to take actions in response to conditions occurring in the system. This is essentially a management application implemented as a resource manager. Using its command-line interface, you can define a condition to monitor. This condition is composed of an attribute to be monitored, and an expression that is evaluated periodically. You also define a response for the condition; the response is composed of zero or more actions and is run automatically when the condition occurs. For more information, see "Basic resource monitoring" on page 136 and "Advanced resource monitoring" on page 152.
- **File system resource manager**, which manages file systems.
- **Host resource manager**, which manages resources related to an individual machine.
- **Least-privilege resource manager**, which enables you to enhance the security, performance, and control of applications that require root authority to run. For more information, see "Controlling access to root commands and scripts" on page 214.
- **Management domain resource manager**, which provides the ability to create and administer a management domain. This is essentially a management application that is implemented as a resource manager. A command-line interface to this resource manager enables you to perform such tasks as creating new management domains, adding nodes to the domain, and listing nodes in the domain. For more information, see "Creating an RSCT management domain" on page 207.

- **Microsensor resource manager**, which provides a way to extend the monitoring capabilities of the system by enabling you to create user-defined attributes for monitoring. For more information, see "Creating event microsensors for monitoring" on page 179.
- **Sensor resource manager**, which provides a means to create a single user-defined attribute to be monitored by the RMC subsystem. For more information, see "Creating event sensor commands for monitoring" on page 177.
- **Storage resource manager**, which provides the ability to protect the data integrity of critical disk resources in an RSCT peer domain. For more information, see "Administering the storage resource manager" on page 239.

For more information about RMC and the core RSCT resource managers, see "Monitoring resources using RMC and resource managers" on page 126.

For information about spooling the resource managers' trace files, see "Configuring, enabling, and controlling trace spooling" in the *RSCT: Diagnosis Guide*.

Cluster security services

RSCT applications and components use cluster security services to perform authentication within both management and peer domains.

Authentication is the process by which a cluster software component, using cluster security services, determines the identity of one of its peers, clients, or an RSCT subcomponent. This determination is made in such a way that the cluster software component can be certain the identity is genuine and not forged by some other party trying to gain unwarranted access to the system. Be aware that authentication is different from authorization (the process of granting or denying resources based on some criteria). Authorization is handled by RMC.

Cluster security services uses *credentials-based authentication*. This type of authentication is used in client and server relationships and enables:

- A client process to present information that identifies the process to the server in a manner that cannot be imitated.
- The server process to correctly determine the authenticity of the information from the client.

Credentials-based authentication involves the use of a third party that the client and the server trust. RSCT supports the host based authentication (HBA) and enhanced host based authentication (HBA2) security mechanisms. For HBA, the UNIX operating system is the trusted third party. HBA is used between RSCT and its client applications, such as TSA and is also used by the configuration resource manager when creating nodes for or adding nodes to an RSCT peer domain.

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

Cluster security services authentication concepts

Authentication is the process by which a cluster software component, using cluster security services, determines the identity of one of its peers, clients, or an RSCT subcomponent.

This determination is made in such a way that the cluster software component can be certain the identity is genuine and not forged by some other party trying to gain unwarranted access to the system. Be aware that authentication is different from authorization (the process of granting or denying resources based on some criteria). Authorization is handled by RMC and is discussed in "Managing user access to resources using RMC ACL files" on page 126.

Cluster security services uses *credentials-based authentication.* This type of authentication is used in client/server relationships and enables:

- a client process to present information that identifies the process to the server in a manner that cannot be imitated.
- the server process to correctly determine the authenticity of the information from the client.

Credentials-based authentication involves the use of a third party that the client and the server trust. RSCT supports the host-based authentication (HBA) and enhanced host-based authentication (HBA2) security mechanisms. For HBA, the UNIX operating system is the trusted third party. HBA is used between RSCT and its client applications, such as TSA and is also used by the configuration resource manager when creating nodes for or adding nodes to an RSCT peer domain.

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

Understanding credentials-based authentication

Credentials-based authentication involves the use of a trusted third party to perform authentication in client/server relationships. To enable this type of authentication, cluster security services provide an abstraction layer between the cluster components and the underlying security mechanisms.

The *mechanism abstraction layer (MAL)* converts mechanism-independent instructions requested by the application into general tasks to be performed by any mechanism. The tasks are carried out by a *mechanism pluggable module (MPM).* An MPM is a component that converts generalized security services routines into the specific security mechanism functions necessary to carry out a request.

Cluster security services currently provides MPMs for these security mechanisms:

- Host based authentication mechanism (HBA), shipped in the file **/usr/lib/unix.mpm**
- Enhanced host based authentication mechanism (HBA2), shipped in the file **/usr/lib/hba2.mpm**

Note: Configuring the HBA2 security mechanism for use in a peer domain is currently not supported.

Additional MPMs to support other security mechanisms may be added in the future.

An MPM configuration file is located on each node of your system in the file **/usr/sbin/rsct/cfg/ctsec.cfg**. The **ctsec.cfg** file defines information about the available MPMs, such as their mnemonics and the full path names of the MPMs. Do *not* modify this file. Be aware, however, that it exists and that cluster security services uses it to locate an MPM.

Understanding the cluster security services private/public key-based mechanisms

Cluster security services provides two security mechanisms for authentication and authorization.

Cluster security services provides two security mechanisms for authentication and authorization:

- The host based authentication mechanism (HBA)
- The enhanced host based authentication mechanism (HBA2)

These mechanisms are shipped in separate modules but make use of the same keys, trusted host lists, and subsystems.

Both the HBA and HBA2 mechanisms employ private/public key pairs for authentication. An unique private/public key pairing is associated with each node in the cluster. These keys are used to encrypt and decipher data. Data encrypted with a particular private key can be deciphered only by the corresponding public key, and data encrypted with the public key can be deciphered only by the corresponding private key.

When the cluster security services are installed on a node, a private key for that node is created and a public key is then derived from the private key. The private key remains on the node in a protected file that only the *root* user can access. The public key is provided to the other nodes within the cluster by the

cluster management applications and is stored on the node in a file that is readable by all users. A node's private/public key pair is considered synonymous with a node's identity and is not expected to change over time. However, if a node's private key does need to be changed, see “Changing a node's private/public key pair” on page 279 for instructions on how to do this.

At the time that the private/public key pair is created, cluster security services also creates a trusted host list for the node. The *trusted host list* associates host identities—host names and IP addresses—with their corresponding public key values. This association is stored in a file and used by the **ctcas** subsystem to create and validate identity-based credentials for the HBA and HBA2 MPMs. This association is also used by RSCT during cluster setup to create and verify signatures for messages passed between RSCT subcomponents.

The initial trusted host list file created by the installation process associates a node's public key value with all active host names and IP addresses associated with all configured and active AF_INET and AF_INET6 adapters for that node. If no network interfaces are configured and active at the time, then the installation process does not create a trusted host list; instead, the **ctcas** subsystem will create the file when it runs for the first time.

A node can only verify credentials from an HBA or HBA2 MPM client if the client's node is listed along with its associated public key in the receiving node's trusted host list. If a client node's host name or IP address does not appear in this list or if the public key value recorded in this list is not correct, then the client cannot be authenticated. When a public key for a cluster node is provided to a node, the public key is recorded along with the cluster node's host name and IP address values in the trusted host list.

Note: If a node's host name or IP address is changed, its private/public key pair does not need to change. You will, however, need to modify the trusted host list file of any node that references the changed node. Specifically, you will need to modify the trusted host list file to include the new host name or IP address, associating it with the existing public key. Also, delete the obsolete host name or IP address from the trusted host list on any node that references it. This is particularly important if the host name or IP address will be reused on another machine. Use the **ctsthl -d -n** {*hostname* | *IP_address*} command to remove obsolete entries.

For message signature verification to function and for HBA and HBA2 credential verification to succeed, the public keys and their host associations must be distributed throughout the cluster. When configuring a cluster of nodes (either as a management domain or as an RSCT peer domain using configuration resource manager commands), the necessary public key exchanges will, by default, be carried out by configuration resource manager utilities. If the network is relatively secure against identity and address spoofing, you can use these utilities; otherwise, transfer the keys manual to prevent the inclusion of nodes that may be attempting to masquerade as known nodes. Carefully consider whether the network's security is sufficient to prevent address and identity spoofing. If you do not think the network is secure enough, see “Guarding against address and identify spoofing when transferring public keys” on page 276. If you are not sure whether your network is secure enough, consult with a trained network security specialist to find out if you are at risk.

Table 3 lists the default locations for a node's private key, public key, and trusted host list.

Table 3. Default locations for a node's private key, public key, and trusted host list files

For these files on a node...	The default location is...
private key	**/var/ct/cfg/ct_has.qkf** This file is readable and accessible only to the *root* user.
public key	**/var/ct/cfg/ct_has.pkf** This file is readable by all users on the local system. Write permission is not granted to any system user.

Table 3. Default locations for a node's private key, public key, and trusted host list files (continued)

For these files on a node...	The default location is...
trusted host list	**/var/ct/cfg/ct_has.thl** This file is readable by all users on the local system. Write permission is not granted to any system user.

Note: You can change the default locations for a node's private key, public key, and trusted host list files by modifying the **ctcasd.cfg** configuration file read by the **ctcas** subsystem upon startup. See "Configuring the ctcasd daemon on a node" on page 270 for more information on properly updating this file and informing cluster security services of the change.

The host based authentication mechanism (HBA):

The host based authentication mechanism (HBA) is configured by default to be the preferred security mechanism.

If all systems in the cluster support both the HBA and the HBA2 MPMs, the default configuration will select the HBA mechanism for authentication processing. Currently, this mechanism is the only MPM supported for use on Hardware Management Consoles (HMCs). The HBA MPM does not require time-of-day clock synchronization within the cluster.

The default MPM priorities should not be modified unless all criteria for each MPM are met by the nodes within the cluster. To alter the preference of security mechanisms, see "Configuring the cluster security services library" on page 268.

The enhanced host based authentication mechanism (HBA2):

The enhanced host based authentication mechanism (HBA2) is considered a separate security mechanism by cluster security services and, therefore, can only be used for authentication if both the client system and the server system provide the HBA2 MPM.

While HBA2 provides improved performance and improved credential tracking over HBA, HBA2 requires time-of-day clock synchronization throughout the cluster in order to function correctly. To prevent authentication failures caused by time-of-day clock disagreement between cluster nodes, the HBA2 mechanism is configured by default to be lower in preference to the HBA mechanism.

Currently, the HBA2 MPM cannot be activated as the primary security mechanism for Hardware Management Consoles (HMCs). Therefore, the HBA MPM will continue to be the preferred mechanism for these platforms.

In addition, configuring the HBA2 security mechanism for use in a peer domain is currently not supported.

The default MPM priorities should not be modified unless all criteria for each MPM are met by the nodes within the cluster. To alter the preference of security mechanisms, see "Configuring the cluster security services library" on page 268.

Host name resolution impacts on HBA and HBA2 MPMs:

The private/public key pairs created by the cluster security services are associated with a node's host name.

Usually, authentication will be based on the resolution of a node's host name. For this reason, it is critical that all hosts within the cluster be configured to resolve host names using the same consistent method. Specifically, a node's host name should resolve to the same exact value if the name resolution is

attempted on any node within the cluster. If one node resolves a node's host name to a value that is different than the value obtained from another node, the HBA and HBA2 MPMs can experience authentication failures when those two nodes interact.

If a domain name service (DNS) is in use, all nodes within the cluster should make use of it. All hosts must be configured to provide host names to applications using either short host names or fully-qualified host names (short name plus domain name). If the cluster includes nodes from multiple domains, you must use fully-qualified host names. If this consistency is not enforced, authentication failures can occur between nodes within the cluster.

IP address support in HBA and HBA2 MPMs:

The private/public key pairs created by the cluster security services are also associated with a node's known IP addresses.

Cluster applications can choose to authenticate with the HBA and HBA2 MPMs using IP addresses instead of host names, thereby removing the need to establish host name resolution or ensure consistent host name resolution throughout the cluster.

The ctcas System Resource Controller subsystem:

The HBA and HBA2 mechanisms employ the **ctcasd** daemon to obtain and authenticate identity-based credentials. This daemon operates as the **ctcas** System Resource Controller subsystem and runs with authority of the *root* user.

Client-side applications use the HBA and HBA2 mechanisms to obtain identity information about the application. The **ctcas** subsystem obtains this identity information, encapsulates the information into an identity-based credential, and uses the private key for the node to identify and protect the credential. This credential is returned to the cluster security services client application, which is expected to transmit this credential to the server application as a means of proving the client's identity.

The HBA and HBA2 mechanisms on the server node employ the **ctcas** subsystem to authenticate the credential. The **ctcas** subsystem uses the public key associated with the client system to verify signatures and encryptions used to identify and protect the credential.

The **ctcas** subsystem is started upon demand when a cluster security services client requests a credential or attempts to verify a credential from another source. If no such requests have been made since system startup, the **ctcas** subsystem will be inoperative. Therefore, an inoperative **ctcas** subsystem is not necessarily an indication of a problem; instead, it might be an indication that no authentication or credential requests have been made of this node.

The cluster security services invoke the **ctstrtcasd** utility for a client whenever it receives an HBA or HBA2 client request and detects that the **ctcasd** daemon is not currently active. The **ctstrtcasd** utility is shipped as a set-user-identity-on-execution binary file and allows any system user to start the **ctcas** subsystem. Normally, only system super users can start a System Resource Controller subsystem; however, this utility permits the SRC to start the **ctcas** subsystem in response to an authentication request from any system user.

Cluster security services authorization concepts

Authorization is the process by which a cluster software component grants or denies access to resources based on certain criteria.

The criteria supported by the cluster security services permit a cluster component to permit or deny access based upon:

- The network identity—the identity reported by the authenticating security mechanism—of the requestor
- The mapped identity of the requestor
- The membership of the requestor within a user group

When a service application authenticates a client using cluster security services, the server establishes a *security context* for the client. The security context contains access to the client's network identity, the local user name to which the network identity may be mapped, and any user group associations for that identity.

Currently, the only RSCT component that implements authorization is RMC, which uses access control list (ACL) files to control user access to resource classes and their resource instances. In these ACL files, described in "Managing user access to resources using RMC ACL files" on page 126, you can specify the permissions that users must have in order to access particular resource classes and resources. The RMC component subsystem uses cluster security services to map the operating system user identifiers specified in the ACL file with the network security identifiers that are verified by the cluster security services authentication process to determine if the user has the correct permissions. This is called *native identity mapping* and is described in "Understanding native identity mapping."

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

Understanding typed network identities

Typed network identities are determined by the mechanism pluggable modules (MPMs) invoked by the mechanism abstraction layer (MAL) during the authentication process.

When authentication completes, the MPM reports either the authenticated network identity to the service application's security context or it reports that the party cannot be authenticated. The inability to authenticate a potential client is not necessarily a failure because a service application may decide to grant some basic level of access to all potential clients. The format of the network identity is specific to the MPM that is used to perform the authentication. The network identity's *type* is its association to the MPM that authenticated that identity.

Network identities for potential clients can be used in access control lists to grant or deny access to resources controlled by the service application. If a network identity and its associated MPM are listed in the ACL, then the level of access associated with that typed network identity can be granted.

The handling of typed network identities in authorization can be affected by *alternate authorization mechanisms*. This is discussed further in "Understanding alternate authorization mechanisms" on page 13.

Understanding native identity mapping

The process of mapping operating system user identifiers to network security identifiers is called *native identity mapping* and is performed by the cluster security services *identity mapping service*.

As described in "Understanding typed network identities," the typed network identity of a client is determined by the MPM invoked by the MAL during the authentication process. If the client application has been authenticated, the MPM will report the authenticated network identity to the service application's security context and associate that identity with the MPM that provided it. The MPM and the cluster security services identity mapping service also determine whether this typed network identity has been mapped to an identity that is native to the local operating system. This function permits cluster security services to associate a local operating system user identity to the network identity that is authenticated by the MPM. This is important for later authorization since, in a cluster of nodes, there is no concept of a common user space. In other words, on the different nodes in the cluster, some user names may represent the same user, while other user names may represent different users on different hosts.

The identity mapping service uses information stored in the identity mapping files: **ctsec_map.global** and **ctsec_map.local**. These identity mapping files are text files containing entries that associate operating system user identifiers on the local system with network security identifiers for authorization purposes. Each node of the cluster has a **ctsec_map.global** file (which contains the common, cluster-wide, identity mappings) and, optionally, may have a **ctsec_map.local** file (which contains identity mappings specific to the local node only).

When the RSCT cluster security services are installed on a node, a default **ctsec_map.global** file is installed. This file contains the default, cluster-wide, identity mapping associations required by RSCT components in order for these systems to execute properly immediately after software installation. There is no default **ctsec_map.local** file.

To modify the cluster-wide identity mappings or a local node's identity mappings, see "Configuring the global and local authorization identity mappings" on page 280.

Example: The network identity *zathras@epsilon3.org*, authticated by the HBA MPM, can be associated with the local operating system user *johndoe* through the following entry in the service application node's **ctsec_map.local** file:

```
unix:zathras@epsilon3.org=johndoe
```

The mapped identity will be available through the service application's security context.

Note: A typed network identity will always take precedence over native mapped identities or identity groups in authorization processing.

The mapping of network identities to native operating system identities during authentication can be affected by *alternate authorization mechanisms*. This is discussed further in "Understanding alternate authorization mechanisms" on page 13.

Understanding identity groups

Cluster systems of even moderate size can potentially generate a large number of network identities that would need to be recorded in access control lists in order for service applications to grant or restrict access to various clients.

To make the management of authorization easier, the cluster security services permit network identities to be grouped so that access to resources can be granted or denied based on a network identity's group membership.

The concept of native identity mapping, discussed in "Understanding native identity mapping" on page 11, is critical to the proper functioning of group-based authorization. This feature allows the cluster security services to map the authenticated network identity of the client to a native operating system user identity. Once this mapping has been found, the MAL can query the local operating system for any groups associated with this native user identity.

Access control lists can be constructed to associate these local operating system user groups to levels of access, thereby avoiding the need to list every single network identity of every single potential client in the access control lists. By using groups instead of network identities, access control lists can be greatly simplified for moderate to large RSCT clusters.

Example: The network identity *zathras@epsilon3.org* from the HBA MPM has been mapped to the local operating system user *johndoe* through the **ctsec_map.local** file. To use identity grouping for authorization, the local operating system user *johndoe* must exist and must be associated with at least one user group. In this example, the user *johndoe* has the following group associations listed in the operating system's **/etc/groups** file:

```
rwgroup:!:100:johndoe,jqpublic,rosebud
```

The MAL would find the group *rwgroup* as a user group for the HBA MPM network identity *zathras@epsilon3.org*. This group can then be used in access control lists to grant or deny access for this client instead of its typed network identity.

Note: A typed network identity will always take precedence over native mapped identities or identity groups in authorization processing. If an entry for a typed network identity is located, that entry is used to grant or deny access, even if an entry for the client's group exists in the same access control list.

Understanding alternate authorization mechanisms

A security context is established in the service application for the client application when the client is authenticated.

This is discussed in "Cluster security services authorization concepts" on page 10. Through this security context, the service application can obtain the client application's typed network identity. This type is the association of the network identity to the MPM that authenticated that identity.

Example: If the client identity *zathras@epsilon3.org* was authenticated by the cluster security services using the HBA mechanism, that identity is typed as an HBA MPM identity and the HBA MPM's mnemonic of **unix** is associated with the *zathras@epsilon3.org* identity in the security context.

Unless otherwise instructed, the cluster security services MAL will always treat an authenticated network identity using its type when attempting to resolve any native identity mappings for this identity or when checking for this identity in access control lists. In other words, once a network identity has been typed as belonging to a specific MPM, that network identity will always be treated as if it originated from that MPM. *The only exception to this rule is when an alternate authorization mechanism has been configured for the authenticating MPM.*

An *alternate authorization mechanism* instructs the MAL to treat a typed network identity as if it were authenticated by a *different* MPM when the MAL performs native identity mapping and authorization. Configuring an alternate authorization mechanism can have a profound impact on authorization processing and, therefore, this feature must be clearly understood.

Alternate authorization mechanism behavior is perhaps best demonstrated through an example:

Example: Consider the case where a server node supports both the HBA MPM and the HBA2 MPM. The node is configured to specify the HBA MPM as the alternate authorization mechanism for the HBA2 MPM. The HBA MPM on the same node is not configured to use an alternate authorization mechanism. A client connects to a service running on this node and uses the HBA2 MPM to authenticate to a service application. The HBA2 MPM authenticates the client identity as *ranger1@ialliance.gov* and types this network identity as **hba2** for the HBA2 MPM. The native identity mapping process then attempts to find a map for this network identity. In the **ctsec_map.local** file, the following entries exist:

```
unix:ranger1@ialliance.gov=valen
hba2:ranger1@ialliance.gov=sinclair
```

Normally, the native identity mapping process would associate the user *sinclair* to **hba2**-typed identity *ranger1@ialliance.gov*. However, because this node specifies the HBA MPM (known by its mnemonic of **unix**) as the alternate authorization mechanism, the native identity mapping treats the network identity of *ranger1@ialliance.gov* as if it were a **unix**-typed identity, not an **hba2**-typed identity. Thus, the native identity mapping process obtains the local operating system user identity of *valen*.

During authorization processing, the identity *ranger1@ialliance.gov* would be considered to be a **unix** network identity. Assume that the service application's access control list contains the following entries:

```
1  type: unix  identity: zathras@epsilon3.org   access: rw
2  type: hba2  identity: ranger1@ialliance.gov  access: rwxd
3  type: unix  identity: ranger1@ialliance.gov  access: r
```

Then, the authorization processing would apply the access rules specified in entry **3**, not entry **2**.

Understanding the use of alternate authorization mechanisms:

By using alternate authorization mechanisms, service application owners and administrators can reuse existing access control lists and infrastructures that were built for use in a single-MPM RSCT environment in an environment where multiple MPMs exist.

Prior to the availability of this feature, the service application owner or administrator would need to create additional access control list entries for the potential new MPM identities. This would appear to the service application administrator to be a duplication of information, especially in the cases of the HBA and HBA2 MPMs where network identities appear to be identical.

The use of an alternate authorization mechanism allows the service application owner or administrator to "pretend" that typed network identities from one MPM are really network identities from a different MPM. This permits the reuse of existing access controls for typed network identities, provided that the MPMs generate network identities that appear exactly the same for the same clients.

For example, if access controls were already established in an RSCT cluster assuming that all potential clients would be authenticated using the HBA MPM, a service application administrator might not want to duplicate that same list of identities for the HBA2 MPM. This is possible since the **unix** and **hba2** MPMs generate network identities that appear exactly the same in format and content for the same clients.

Note: Individual service applications do not decide whether they wish to exploit or avoid this feature. Because this feature is enabled through the cluster security services configuration files, all applications using the cluster security services are affected by the configuration of an alternate authorization mechanism.

Setting an alternate authorization mechanism:

The **/usr/sbin/rsct/cfg/ctsec.cfg** configuration file defines the available MPMs.

This is mentioned in "Understanding credentials-based authentication" on page 7. For the MPM entries defined in the **ctsec.cfg** file, the MAL provides a **z** flag that an MPM can use to specify the alternate authorization mechanism to use for this mechanism. This provides the capability to allow separate mechanisms for authentication and authorization, as discussed earlier. The **z** flag takes a required attribute that specifies the MPM to be used for authorization, as follows:

- If the **z** flag is *not* present on an MPM entry in the **ctsec.cfg** file, then the authorization mechanism to be used is the MPM's own authorization mechanism, specified by the mnemonic of the MPM entry. That is, both the authentication mechanism and the authorization mechanism of the same MPM are to be used.
- If the **z** flag *is* present on an MPM entry in the **ctsec.cfg** file, then the authorization mechanism to be used is the mechanism that is used by the MPM specified by the **z** flag's attribute. That is, the authentication mechanism is that of the MPM defined by the configuration record in **ctsec.cfg** while the authorization mechanism is that of the MPM specified by the **z** flag's attribute.

The enhanced host based authentication (HBA2) MPM supports the use of the **z** flag to specify a separate authorization mechanism; the host based authentication (HBA) MPM does not.

Topology Services subsystem

The *Topology Services subsystem* (or *Topology Services*) provides other RSCT applications and subsystems within an RSCT peer domain with network adapter status, node connectivity information, and a reliable messaging service.

The Topology Services subsystem runs as a separate daemon process on each machine (node) in the peer domain. The adapter and node connectivity information is gathered by these instances of the subsystem forming a cooperation ring called a *heartbeat ring*. In the heartbeat ring, each Topology Services daemon process sends a heartbeat message to one of its neighbors and expects to receive a heartbeat from another. In this system of heartbeat messages, each member monitors one of its neighbors. If the neighbor stops responding, the member that is monitoring it will send a message to a particular Topology Services daemon that has been designated as a *group leader*.

In addition to heartbeat messages, each Topology Services daemon process sends connectivity messages around all heartbeat rings. Each ring forwards its connectivity messages to other rings so that all nodes can construct a connectivity graph. The reliable messaging service uses this graph to determine the route to use when delivering a message to a destination node.

Related information:

"The Topology Services subsystem" on page 286
In an RSCT peer domain, the configuration resource manager uses the Topology Services subsystem to monitor the liveness of the adapters and networks included in communication groups.

Group Services subsystem

The *Group Services subsystem* (or *Group Services*) provides other RSCT applications and subsystems within an RSCT peer domain with a distributed coordination and synchronization service.

The Group Services subsystem runs as a separate daemon process on each machine (node) in the peer domain. A group is a named collection of processes. Any process may create a new group, or join an existing group, and is considered a Group Services client. Group Services guarantees that all processes in a group see the same values for the group information and that they see all changes to the group information in the same order. In addition, the processes may initiate changes to the group information.

A client process may be a *provider* or a *subscriber* of Group Services. *Providers* are full group members and take part in all group operations. *Subscribers* merely monitor the group and are not able to initiate changes in the group.

Related information:

"The Group Services subsystem" on page 308
The configuration resource manager uses the Group Services subsystem to provide distributed coordination, messaging, and synchronization among nodes in an RSCT peer domain.

System resource controller

The system resource controller (SRC) provides a set of commands and subroutines to make it easier for the system manager and programmer to create and control subsystems.

A subsystem is any program or process or set of programs or processes that is usually capable of operating independently or with a controlling system. A subsystem is designed as a unit to provide a designated function. Specifically, the RSCT subsystems, such as Topology Services, Group Services, and cluster security services, run under the SRC. On AIX, the SRC is, like the RSCT components, part of the operating system. For the Linux, Solaris, and Windows implementations of RSCT, the SRC is packaged with the RSCT components.

The SRC was designed to minimize the need for operator intervention. While it provides a consistent user interface for starting subsystems, stopping subsystems, and performing status inquiries, its operation should be largely transparent to you. Do not explicitly start or stop the RSCT subsystems under normal circumstances. An IBM Service representative might instruct you to use the SRC commands **startsrc** and **stopsrc** to start and stop RSCT subsystems when following certain troubleshooting procedures. You can also use the command **lssrc** to list the status of RSCT subsystems.

RSCT network considerations

Make these considerations when running RSCT components.

In order for RSCT components to run on and communicate over a network, be aware of the following:

- All network interfaces attached to the same subnet over which RSCT traffic may pass must have the same MTU size.
- The port numbers and service definitions used by the various RSCT components.
- When the route striping is disabled, and the two network interfaces are on the same subnet, pings might never be sent across if the interface that owns the subnet route fails.

Network configuration

All network interfaces attached to the same subnet over which RSCT traffic may pass must be configured to use the same MTU (maximum transmission unit) size.

This is important because MTU size differences between nodes may result in packet loss, making it impossible for RSCT components to communicate with their peers on other nodes.

The **netstat -i** command will display the MTU size for network interfaces that are currently configured on a node. To set the MTU size, you can use the **ifconfig** command. For more information on the **netstat** and **ifconfig** commands, see their online man pages.

IPv6 addresses

IPv6 addresses can be made visible as resources in the **IBM.NetworkInterface** class and used for heartbeating and internal peer domain operations, as controlled by the **IPv6Support** attribute.

For each interface that has IPv6 addresses configured on it, one of these IPv6 addresses will be chosen for representation as a resource, if the **IPv6Support** attribute is set to **1** (True). If global addresses are configured on an interface in addition to its link-local address, the first global address (as displayed by **ifconfig -a**, for example) will be represented in preference to the link-local address.

All such interfaces represented as resources will be formed into communication groups with full end-to-end connectivity as supported by physical networking and routing. Communication groups that contain IPv6 interfaces will be disjoint from communication groups that contain IPv4 interfaces. By default, all such IPv6 interfaces will be used for heartbeating and internal peer domain operations, under control of their **HeartbeatActive** attribute and based on characteristics of their underlying IP interfaces. In this regard, peer domain usage of IPv6 interfaces is identical to that of IPv4 interfaces.

Note: The configuration resource manager is set to ignore link-local IPv4 and IPv6 addresses from being taken into account for heartbeat, trusted host list (THL), and the **IBM.NetworkInterface** resource class.

RSCT port usage

The tables comprising this topic summarize the service definitions for the RSCT components.

For each service definition, the following information is shown:

Service name
: Specifies the service name. The service names shown are examples of how the names may appear in the **/etc/services** file.

Port number
: Specifies the port number used for the service.

  A value with an asterisk (*) as a suffix indicates that the port number can be customized, and that the default is shown.

Protocol name
: Specifies the transport protocol used for the service.

Source port range
A range of port numbers used on either the client side or daemon (server) side of the service. A value of LB indicates that the source port range value should be left blank. In other words, no source port range value should be specified.

Required or optional
Whether or not the service is required.

Description
A short description of the service.

Table 4 describes the service definition for the Topology Services subsystem.

Table 4. Service definitions for RSCT: Topology Services

Service name	Port number	Protocol name	Source port range	Required or optional
cthats	12347*	UDP	1024-65535	required

Description: Network services used by Topology Services for daemon to daemon communication. Note that any firewall rules should allow BROADCAST packets to go through the *cthats* port.

This table shows the default port number. You can customize the port number when issuing the **mkrpdomain** command (as described in "Creating a peer domain definition" on page 52).

The Topology Services port is dynamically added in **/etc/service**, and is only present when the node is online in the peer domain.

Table 5 describes the service definition for the Group Services subsystem.

Table 5. Service definitions for RSCT: Group Services

Service name	Port number	Protocol name	Source port range	Required or optional
cthags	12348*	UDP	1024-65535	required

Description: Network services used by Group Services for daemon to daemon communication.

This table shows the default port number. You can customize the port number when issuing the **mkrpdomain** command (as described in "Creating a peer domain definition" on page 52).

The Group Services port is dynamically added in **/etc/service**, and is only present when the node is online in the peer domain.

Table 6 describes the service definitions for the Resource Monitoring and Control subsystem.

Table 6. Service definitions for RSCT: Resource Monitoring and Control

Service name	Port number	Protocol name	Source port range	Required or optional
rmc	657	UDP	LB	required
rmc	657	TCP	LB	required

RMC network port usage, data flows, and security

The Resource Monitoring and Control (RMC) subsystem is a generalized framework for managing, monitoring, and manipulating resources (physical or logical system entities).

RMC runs as a daemon process on individual machines. You can use it to manage and monitor the resources of a single machine, or you can use it to manage and monitor the resources of a cluster's peer domain or management domain. In a peer domain or management domain, the RMC daemons on the various nodes work together to enable you to manage and monitor the domain's resources.

A peer domain is a set of nodes that have a consistent knowledge of the existence of each other and of the resources shared among them. On each node within the peer domain, RMC depends on a set of core cluster services, which include Topology Services, Group Services and cluster security services. See "Administering an RSCT peer domain" on page 35 for more information.

The term *management domain* is defined as a set of nodes whose resources can be managed and monitored from one of the nodes, which is designated as the management control point (MCP). All other nodes are considered to be managed nodes. Topology Services and Group Services are not used in a management domain. Management domains are created automatically on POWER4, POWER5, POWER6®, and POWER7® systems, one for each HMC that manages the system. Each operating system image is a managed node in each of the domains and the HMCs are the MCPs of the domains. In addition, CSM creates a management domain and xCAT optionally creates a management domain. Each cluster node is a managed node in the domain and the management server is the MCP of the domain.

The term *RMC client application* is defined as a program that connects to an RMC daemon on an individual machine, an MCP, or a node within a peer domain to manage or monitor resources.

This information helps you determine the impact of network firewall on the RMC subsystem and any RMC client application.

Port usage:

The RMC daemon uses TCP port 657 to accept requests from RMC client applications, such as CSM and the various RMC commands that are running remotely from the node upon which the RMC daemon is running.

The RMC commands are documented in the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides. The RMC daemon binds to this port only when enabled to accept remote client connections. The daemon is automatically enabled to accept remote connections when CSM is installed on the node, when xCAT optionally configures the node for RMC monitoring, or if the node is an operating system image in a POWER4, POWER5, POWER6, or POWER7 system. The **rmcctrl** command enables the daemon to accept remote client connections or disables the daemon from accepting remote client connections. Note that disabling this function can result in failure of applications expecting to remotely connect to an RMC daemon.

The maximum number of remote connections outstanding to an RMC daemon at any time can be no more than 232.

The RMC daemon also uses the UDP port 657 to communicate with all other RMC daemons in the same domain. When the daemon starts, it binds to this port, provided the node upon which the daemon is running is configured to be an MCP, a managed node, or a peer node. Otherwise, it binds to this port when the node is configured into a domain. Note that the same node can be online in a peer domain, be an MCP of one management domain, and be a managed node in one or more other management domains, simultaneously.

When binding to these ports, the daemon does so in such a manner that packets may be received on any available interface.

RMC client applications use an ephemeral TCP port assigned by the operating system.

Data flow over TCP/IP:

TCP is a connection oriented, reliable stream protocol. As such, the RMC client application always initiates the connection to an RMC daemon and the client/daemon messages are of varying length.

The actual size and number of packets per message at the network level is a function of the TCP/IP implementation. Message retries due to errors in the network are handled by TCP.

Once the connection is established at the network level, the client initiates a start session protocol with the daemon. This protocol consists of a series of message exchanges, where the client sends a start session message to the daemon and the daemon returns a start response message to the client. The purpose of the start session protocol is to determine the functional level of the other end of the connection, mutually authenticate the client and the RMC daemon, and to determine if message authentication is to be enabled. Note that the client can specify that the session is to be unauthenticated; if so, message authentication is not enabled.

After the start session protocol is complete, the client sends one or more command messages to the daemon over a period of time and at an interval determined by the client application. For each command message, there are one or more response messages. The number of response messages is a function of the command. The time between the command and its responses is a function of the command and system and network loads. Note that the client may have a number of commands outstanding at any time and responses are ordered relative to the command for which they are being returned, not any other command. In other words, the responses for command **N+1** may be returned prior to, or intermixed with, the responses for command **N**. Finally, a command that is an event registration request results in zero or more event notification responses, asynchronous to any other command.

A session remains open until it is terminated by the client application or, in certain error situations, by the RMC daemon. Termination of the client application or the RMC daemon always results in termination of the session.

Data flow over UDP:

To achieve scaling and configuration goals, the RMC daemon employs the UDP protocol to communicate with other RMC daemons in the domains in which it is a member.

Since the UDP protocol is not reliable itself, the RMC daemon handles message retries. Messages are varying in length, but are never more than 4546 bytes. The actual size and number of packets per message at the network level is a function of the TCP/IP implementation.

There are six basic types of messages used by RMC for daemon-to-daemon communication:

- Heartbeat (Hb)
- Heartbeat Acknowledgement (HbA)
- Synchronize (Syn)
- Synchronize Acknowledgement (SynA)
- Data (Data)
- Data Acknowledgement (DataA)

Hb and HbA messages are only used within a management domain to establish and maintain a communication channel between the RMC daemon on the MCP and the RMC daemon on each managed node. Hb messages are sent by the MCP to each managed node and each managed node returns a HbA. A Hb is sent to each managed node every 12 seconds (by default). It is not necessary that there be a matching HbA for every Hb, but if a HbA is not received from a managed node after sending 8 (by default) Hb messages to it, the MCP considers that managed node to be down. It continues to send Hb messages to the managed node at 12 second intervals. Once the managed node again responds to the Hb messages, the managed node is declared up. Given this, the default heartbeat timeout interval is 96 seconds.

If a managed node has not received a Hb within the heartbeat timeout interval, it declares the MCP down and will start sending Hb messages to the MCP. Once the managed node has received a HbA or a Hb it stops sending Hb messages to the MCP and declares the MCP up. Given such factors as network

conditions, whether a node is powered on or off, the state of the operating system and the state of the RMC daemon itself, it is possible that the Hb message may be initiated by either the MCP or the managed node.

In a peer domain, the RMC daemon relies on the Topology Services and Group Services components to indicate when a communication channel is established between any two nodes and to indicate when a node is to be declared as up or down. In either a management domain or a peer domain, declaring that a node in the domain is down results in recovery actions including, but not limited to, returning errors for any outstanding commands to that node and dropping any responses to that node. Declaring a node is up results in recovery actions including, but not limited to, reestablishing event registrations that are applicable to that node.

Once the communication channel is established between two RMC daemons in a domain, and prior to the exchange of any data messages, data synchronization is established using the Syn message and the SynA message. For each Syn message that is sent, there is a matching SynA message response. These messages are exchanged until synchronization is achieved. The Syn message may be initiated from either the MCP or the managed node in a management domain or from either of any two pairs of nodes in a peer domain, depending on which RMC daemon has data messages queued for transmission. After a node is declared down and then back up, data synchronization must be reestablished.

Data messages are used to send commands from RMC client applications connected to the RMC daemon on an MCP or peer node to the appropriate nodes within the domain. Responses to the commands are returned via Data messages. Given the maximum size of a message specified above, several Data messages may be required to transmit all of a command or a response. For each Data message there is a matching DataA message; a Data message is not successfully transferred until its matching DataA message is received. If a command or response requires several Data messages, the **Nth+1** Data message is not sent until the **Nth** Data message is successfully transferred.

The relationship of commands and responses between any two RMC daemons is the same as described in "Data flow over TCP/IP" on page 18, although a command may originate from any client that is connected to a daemon, and the response is returned to the appropriate client connected to the daemon. In addition to commands, responses and event notifications, Data messages are used to exchange information that is internal to the RMC daemons for domain configuration. Such information is typically sent to a node as soon as it has been declared up.

Security:

By default, message authentication is employed for the TCP RMC protocols and the UDP RMC protocols. In the TCP case, it is used to authenticate messages exchanged between an RMC client application and an RMC daemon.

Message authentication guarantees that a message received by the RMC daemon came from the authenticated client that started the session and that a message received by the client application came from the authenticated RMC daemon with which it started the session. Message authentication is not employed between unauthenticated client applications and the RMC daemon; unauthenticated clients must have specific authorization to access RMC resources, as specified in an RMC ACL. In the UDP case, it is used to authenticate messages between any two RMC daemons in a management domain. Message authentication guarantees that the daemon receiving a message knows it came from the sending daemon that is identified in the message. Currently, message authentication is not supported between two RMC daemons in a peer domain.

Message authentication is based on a host's (node's) public/private key pair. Public keys are automatically exchanged within a domain when the domain is created or when nodes are added to the domain. In a management domain, public keys are exchanged between the MCP and each managed node, but not between managed nodes. In a peer domain, public keys are exchanged among all nodes that are defined in the peer domain.

When authentication is enabled, any message that cannot be authenticated is dropped by the receiver without any notification to the sender. If authentication is not enabled, any message that does not conform to the RMC proprietary protocol is dropped by the receiver without any notification to the sender. Furthermore, if the nonconforming message is from an RMC client application, the daemon closes the connection. In any case, the RMC protocols only permit the sender to execute predefined RMC commands.

The **rmcctrl** command can disable message authentication or require that it must be used.

When message authentication is enabled (the default), it means that, provided both ends of the connection have the code to do message authentication, it will be turned on. The only way to turn message authentication off is to set the policy to disabled. *Required* means that if the other end of the connection does not have the code to do message authentication or it is disabled, then no communication is allowed.

TCP message authentication is available on:

- RSCT 3.1.0.0, or later for AIX 7.1
- RSCT 2.5.2.0, or later for AIX 6.1 and Linux

UDP message authentication is available on:

- RSCT 3.1.0.0, or later for AIX 7.1
- RSCT 2.5.2.0, or later for AIX 6.1 and Linux

Management domain configuration:

When an operating system image is started on a POWER4, POWER5, POWER6, or POWER7 system, the resource monitoring and control (RMC) subsystem is started and then, the **IBM.CSMAgentRM** or **IBM.MgmtDomainRM** resource manager is started. These resource managers implement the necessary functions to automatically add the operating system image, as a managed node, to the management domains created automatically on each Hardware Management Console (HMC). The **IBM.MgmtDomainRM** resource manager is started on AIX 7.1. It is also started on AIX 6.1 if **csm.client** 1.7.1.4 (PTF U834202), or later, is installed. Otherwise, the **IBM.CSMAgentRM** resource manager is started on AIX 6.1. On Linux logical partitions (LPAR) on Power Systems servers, if CSM 1.7.1.6, or later (or no CSM is installed), and RSCT 3.1.0.0, or later, are installed, the **IBM.MgmtDomainRM** resource manager is started. Otherwise, for previous versions of CSM or RSCT, the **IBM.CSMAgentRM** resource manager is started.

When the logical partition (LPAR) is managed by the **IBM.MgmtDomainRM** resource manager, resources of the **IBM.ManagementServer** class are replaced by resources of the **IBM.MCP** class.

The following persistent attributes of the **IBM.ManagementServer** class are supported by the **IBM.MCP** resources:

Table 7. Supported persistent attributes for IBM.MCP resources

Persistent attribute	Description
MNName	The primary host name of the managed node as known to the management control point (MCP) (string).
NodeID	The RSCT node ID of the MCP (uint64).
KeyToken	The host identity that is used to look up the public key of the MCP in the trusted host list (THL) on the managed node (string).
IPAddresses	The list of IP addresses that are assigned to the MCP and reachable from the managed node. Each address must be on a separate subnet (string array).
ConnectivityNames	The list of IP addresses that are assigned to the node and reachable from the MCP (string array). One of these addresses is also the value of the **MNName** attribute.

Table 7. Supported persistent attributes for IBM.MCP resources (continued)

Persistent attribute	Description
HMCName	The value of the `HscName` field that is found in the runtime abstraction services (RTAS) (string).
HMCIPAddr	The value of the `HscIPAddr` field that is found in the RTAS (string).
HMCAddIPs	The value of the `HscAddIPs` field that is found in the RTAS (string).
HMCAddIPv6s	The value of the `HMCAddIPv6s` field that is found in the RTAS (string).

However, the following attribute names of the **IBM.ManagementServer** class are not supported by the **IBM.MCP** resources:

- Name
- Hostname
- ManagerType
- LocalHostname
- MgmtSvrNodeID
- MgmtSvrPrimaryNames
- ClusterTM
- ClusterSNum
- HAState
- MgmtSvrHostNamesList
- LocalHostNamesList

If the Integrated Virtualization Manager (IVM) is managed by the **IBM.MgmtDomainRM** resource manager, the resources of the **IBM.ManagedNode** class are replaced with the **IBM.MngNode** resources.

The following persistent attributes of the **IBM.ManagedNode** class are supported by the **IBM.MngNode** resources.

Table 8. Supported persistent attributes for IBM.MngNode resources

Persistent attribute	Description
Name	The primary host name of the managed node (string).
NodeID	The RSCT node ID of the node (uint64).
KeyToken	The host identity that is used to look up the public key of the managed node in the THL on the MCP (string).
Aliases	The additional names for the node; might be NULL (string array).
IPAddresses	The list of IP addresses that are assigned to the managed node and reachable from the MCP. Each address must be on a separate subnet (string array).

However, the following attribute names of **IBM.ManagedNode** class are not supported by the **IBM.MngNode** resources:

- Hostname
- UniversalId
- ConnectivityIPAddrs

These resource managers initiate a series of sessions to the RMC daemon on each HMC, as described in "Security" on page 20:

- A session to each configured IP address of each HMC to validate the IP address. No commands are sent if **IBM.CSMAgentRM** is used. Otherwise, **IBM.MgmtDomainRM** sends few commands.
- A session to each HMC. A command is sent to determine the code level on the HMC of the **IBM.DMSRM** resource manager.

- A session to each HMC. A command is sent containing necessary information, including the node's public key. The response contains necessary information, including the HMC's public key.

As a result of these sessions, barring any errors, the operating system image becomes a managed node, and the RMC daemon begins communication with the RMC daemon on each HMC. Periodically, if the **IBM.CSMAgentRM** resource manager is started, it examines the state of the operating system image and, if appropriate, initiates another series of sessions and commands to the RMC daemon on each HMC to adjust the managed node configuration within the management domains. If the **IBM.MgmtDomainRM** resource manager is started, it checks whether the **IBM.CSMAgentRM** resource manager performed any necessary configuration operations. If so, the **IBM.MgmtDomainRM** resource manager terminates itself (assuming it is not processing any other RMC commands). Then, periodically, the RMC daemon examines the state of the operating system image and, if it changes, restarts **IBM.MgmtDomainRM** to perform any necessary configuration.

In a CSM cluster, when the **updatenode** command is run on a management server, the **dsh** command is used to run the **mgmtsvr** command on each node that is specified to the **updatenode** command. The **mgmtsvr** command then triggers the **IBM.CSMAgentRM** resource manager on the node to initiate a series of sessions to the RMC daemon on the management server:

- A session to the management server. A command is sent to determine the code level on the management server of the **IBM.DMSRM** resource manager.
- A session to the management server. A command is sent containing necessary information, including the node's public key. The response contains necessary information, including the HMC's public key.

As a result of these sessions, barring any errors, the node becomes a managed node. After it becomes a managed node, the RMC daemon begins communication with the RMC daemon on the management server.

If xCAT optionally configures a management domain, the **moncfg rmcmon** command creates an **IBM.MngNode** resource on the management server (MCP) for each xCAT managed node. Then it creates an **IBM.MCP** resource on each managed node. The last step in creating an **IBM.MCP** resource is to run the Public Key Exchange protocol between the **IBM.MgmtDomainRM** resource manager on the MCP and the **IBM.MgmtDomainRM** resource manager on the managed node. As a result of these actions, barring any errors, each node becomes a managed node. After it becomes a managed node, the RMC daemon begins communication with the RMC daemon on the management server.

Ephemeral ports

As with many client/server models, once a client connection has been established with an RSCT subsystem daemon (such as an RMC daemon), the client communicates with the daemon on an ephemeral port in order for the daemon to accept new requests on its base port. Blocking ephemeral port ranges may cause an application that has established a connection with a daemon to fail.

Ephemeral port ranges vary by operating system. On AIX nodes, the following command can be used to determine ephemeral port ranges:

```
/usr/sbin/no -a | fgrep ephemeral
```

The following output is displayed:

```
tcp_ephemeral_low = 32768
tcp_ephemeral_high = 65535
udp_ephemeral_low = 32768
udp_ephemeral_high = 65535
```

On AIX nodes:

- There is an ephemeral port range for TCP and one for UDP. They can overlap, or be the same range.
- The low values cannot be set to anything less than 1024.

- If a free port in the ephemeral range cannot be obtained, the bind or connect operation will report a failure back to the application.

On Linux nodes, the following command can be used to determine ephemeral port ranges:

```
cat /proc/sys/net/ipv4/ip_local_port_range
```

The following output is displayed:

```
32768   61000
```

The preceding AIX and Linux commands display operating system settings.

The range of allowable ephemeral port values might also be affected by applications that support ephemeral port customization. For example, a firewall application may allow ephemeral port ranges to be defined on a per-service basis.

RSCT uses the ephemeral port values supplied by the operating system.

RSCT on Microsoft Windows

Starting with version 2.5.0.0, RSCT supports IBM Tivoli System Automation for Multiplatforms on the Microsoft Windows platform.

In addition to general RSCT administration information, you also need to understand:

- how to verify the installation to ensure that RSCT will run successfully.
- how RSCT handles security authentication and authorization in a Windows environment.

RSCT security considerations on the Windows platform

RSCT's cluster security services provide the security infrastructure that enables components of RSCT to authenticate and authorize the identity of other parties.

RSCT's cluster security services provide the security infrastructure that enables components of RSCT to authenticate and authorize the identity of other parties, as follows:

- *Authentication* is the process of ensuring that a party is who it claims to be. Using cluster security services, various cluster applications can check that other parties are genuine and are not attempting to gain unwarranted access to the system.
- *Authorization* is the process by which a cluster software component grants or denies access to resources based on certain criteria. The only RSCT component that implements authorization is RMC, which uses access control list (ACL) files to control user access to resource classes and to instances of those resource classes.

RSCT cluster security services are disabled when running on a Windows platform. On the Windows platform, authentication is verified by the standard Windows login process and authorization is verified by the standard Windows file permissions on the RSCT commands. A logged in Windows user must have execute permission on the appropriate files in order to run RSCT commands.

All of Tivoli System Automation and RSCT is protected by file permissions. Therefore:

- On the local machine, if a user has the execute file permission on RSCT commands, the user can run RSCT commands. In particular, any user who is an administrator can execute RSCT commands.
- Any user who has remote access to the Windows machine and who has the execute file permission on RSCT commands can run RSCT commands.

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

Verifying RSCT installation

There are AIX, Linux, Solaris, and Windows implementations of RSCT.

The AIX implementation of RSCT is included as part of the IBM AIX operating system. The Linux, Solaris, and Windows implementations of RSCT are included as part of IBM Tivoli System Automation for Multiplatforms.

This documentation applies to RSCT version 3.1.4.0 and later for the AIX 6.1, AIX 7.1, Linux, Solaris, and Windows operating systems.

Verifying RSCT installation on AIX nodes

RSCT installation verification on AIX nodes involves the following information.

To verify that RSCT has been installed on an AIX node, enter:

```
lslpp -L rsct.*
```

For example, for RSCT 3.1.5.0 on AIX 7.1, the system displays the following information:

```
Fileset                      Level  State  Type  Description (Uninstaller)
----------------------------------------------------------------------------
rsct.basic.hacmp           3.1.5.0    C     F    RSCT Basic Function (HACMP/ES
                                                  Support)
rsct.basic.rte             3.1.5.0    C     F    RSCT Basic Function
rsct.basic.sp              3.1.5.0    C     F    RSCT Basic Function (PSSP
                                                  Support)
rsct.compat.basic.hacmp    3.1.5.0    C     F    RSCT Event Management Basic
                                                  Function (HACMP/ES Support)
rsct.compat.basic.rte      3.1.5.0    C     F    RSCT Event Management Basic
                                                  Function
rsct.compat.basic.sp       3.1.5.0    C     F    RSCT Event Management Basic
                                                  Function (PSSP Support)
rsct.compat.clients.hacmp  3.1.5.0    C     F    RSCT Event Management Client
                                                  Function (HACMP/ES Support)
rsct.compat.clients.rte    3.1.5.0    C     F    RSCT Event Management Client
                                                  Function
rsct.compat.clients.sp     3.1.5.0    C     F    RSCT Event Management Client
                                                  Function (PSSP Support)
rsct.core.auditrm          3.1.5.0    C     F    RSCT Audit Log Resource
                                                  Manager
rsct.core.errm             3.1.5.0    C     F    RSCT Event Response Resource
                                                  Manager
rsct.core.fsrm             3.1.5.0    C     F    RSCT File System Resource
                                                  Manager
rsct.core.gui              3.1.5.0    C     F    RSCT Graphical User Interface
rsct.core.hostrm           3.1.5.0    C     F    RSCT Host Resource Manager
rsct.core.lprm             3.1.5.0    C     F    RSCT Least Privilege Resource
                                                  Manager
rsct.core.microsensor      3.1.5.0    C     F    RSCT MicroSensor Resource Manager
rsct.core.rmc              3.1.5.0    C     F    RSCT Resource Monitoring and
                                                  Control
rsct.core.sec              3.1.5.0    C     F    RSCT Security
rsct.core.sensorrm         3.1.5.0    C     F    RSCT Sensor Resource Manager
rsct.core.sr               3.1.5.0    C     F    RSCT Registry
rsct.core.utils            3.1.5.0    C     F    RSCT Utilities
rsct.exp.cimrm             3.1.5.0    C     F    CIM Resource Manager
rsct.opt.storagerm         3.1.5.0    C     F    RSCT Storage Resource Manager
rsct.opt.stackdump         3.1.5.0    C     F    RSCT Stack Dump Function
```

If the RSCT components are installed, make sure they are at a version that applies to this documentation. If you are using RSCT in conjunction with an exploiter of the technology, make sure this is the version of

RSCT the exploiter requires. Not all of the RSCT filesets are required by every RSCT exploiter. See the specific RSCT exploiter's documentation for information about RSCT version and fileset requirements.

The RSCT for AIX filesets are described in Table 9.

Table 9. RSCT for AIX filesets

This fileset...	Contains...
rsct.basic.rte	• Configuration resource manager • Group Services • Topology Services
rsct.core	RSCT core components, including: • Audit log resource manager • Cluster security services • Event response resource manager (ERRM) • File system resource manager • Host resource manager • Least-privilege resource manager • Management domain resource manager • Microsensor resource manager • Resource monitoring and control (RMC) subsystem • Sensor resource manager • System registry • Miscellaneous utilities
rsct.exp	RSCT Expansion Pack, which contains the CIM resource manager. This fileset is part of the AIX Expansion Pack and will not be installed by default. To use the CIM resource manager on AIX, you will also need to install the Pegasus CIM Server filesets off the AIX Expansion Pack. The filesets for the Pegasus CIM server are: sysmgt.pegasus.cimserver sysmgt.pagasus.osbaseproviders
rsct.opt.storagerm	Storage resource manager
rsct.opt.stackdump	Stack dump function

If entering the **lslpp** command as described above reveals that needed RSCT filesets are not installed, you can install them from the AIX installation media using the **installp** command. Enter the **installp** command as shown below, where *cd0* is the name of the AIX installation media, and *fileset* is the name of an RSCT fileset as shown in the preceding table.

```
installp -agXd /dev/cd0 fileset
```

If you are upgrading from a previous release of RSCT, make sure you review the information in "Migration considerations" on page 47 first.

RSCT adapter support for AIX

RSCT for the AIX operating system supports a number of adapters.

RSCT for AIX supports these adapters:

- 10/100 Ethernet adapter.
- 1 GB Ethernet adapter.
- 10 GB Ethernet adapter.
- SP-Switch adapter High Performance Switch adapter.
- SP-Switch2 adapter.
- High Performance Switch adapter.
- 1 GB Myrinet switch adapter.
- Topspin adapter over InfiniBand. For more information when planning for your network, see "IP addressing on an InfiniBand network" on page 32.

- GX Dual-port 4x InfiniBand Host Channel Adapter (HCA). For more information when planning for your network, see “IP addressing on an InfiniBand network” on page 32.
- IBM Token Ring adapter.
- Virtual LAN and Shared Ethernet adapters.
- IP Over Fiber Channel adapters.
- Virtual IP address support.
- Etherchannel configuration support (arrangement in which two or more network interfaces on a host computer are combined for redundancy or increased throughput).
- ATM and FDDI adapters (supported only in a PowerHA configuration).

Verifying RSCT installation on Linux nodes

RSCT installation verification on Linux nodes involves the following information.

The Linux implementation of RSCT is shipped with a number of products that use this technology. The RSCT packages must be installed by following the specific exploiter's installation procedure. Table 10 lists the packages that are required on the target node before installing RSCT.

Table 10. Library and packages

Library	Package
Standard C Library	glibc
Standard C++ Library	libstdc++
Compatibility Standard C++ Library on Red Hat Enterprise Linux (RHEL) 5	compat-libstdc++-33-3.2.3 The RHEL 5 64-bit kernel (**ppc64** and **x86_64**) includes two RPMs for compat-libstdc++-33-3.2.3. One RPM is 32-bit and the other is 64-bit. RSCT requires the 32-bit RPM by default. If you plan to use the RSCT 64-bit library, the 64-bit compat-libstdc++-33-3.2.3 RPM is also required.
Compatibility Standard C++ Library on RHEL 6 (6.1, 6.2, 6.3, and 6.4)	compat-libstdc++-33-3.2.3 RSCT requires libstdc++-4.4.7-3.el6 (32-bit and 64-bit) as base RPMs to install compat-libstdc++-33-3.2.3 on 64-bit kernel.
Compatibility Standard C++ Library on SUSE Linux Enterprise Server 10	compat-libstdc++-5.0.7 or libstdc++33
Compatibility Standard C++ Library on SUSE Linux Enterprise Server 11 (Service Pack 1, Service Pack 2, and Service Pack 3)	libstdc++33-3.3.3-11.9 RSCT requires libstdc++6-4.7.2 (32-bit and 64-bit) as base RPMs to install the libstdc++33-3.3.3-11.9 on 64-bit kernel.
Code set Conversion Library on SUSE Linux Enterprise Server	glibc, glibc-locale, glibc-locale-32bit, or glibc-locale-64bit

RSCT, Linux, and code set conversion

Much of RSCT, particularly the resource monitoring and control (RMC) subsystem, converts strings between the encoding of the client's locale and an encoding of Unicode. Unicode is used internally within the RMC subsystem to store and process strings.

For code set conversion, RMC relies on support that is provided by the underlying operating system. That support must be installed for RMC to function correctly.

The following error messages indicate that the required code set support is not installed:

```
2610-657 System does not support codeset conversion from source_codeset to target_codeset.

This system does not support codeset conversion that RMC needs. Installation will stop.
Please refer to the RSCT Administration Guide for additional RPMs that are required.
```

On Linux, the RPMs that support code set conversion vary depending on distribution, release, and machine type.

The RPMs that provide support for code set conversion for various configurations follow. Notice the difference in RPMs between Red Hat and SUSE. Also, note the change in how 32-bit processes are supported on 64-bit machines using SUSE.

For Red Hat Enterprise Linux Server release 5.3, the RPM is:

```
glibc
```

For SUSE Linux Enterprise Server 10 or 11 running on a 32-bit machine, the RPM is:

```
glibc-locale
```

For 32-bit applications running on SUSE Linux Enterprise Server 10 on a 64-bit machine, the RPM is:

```
glibc-locale
```

For 64-bit applications running on SUSE Linux Enterprise Server 10 on a 64-bit machine, the RPM is:

```
glibc-locale-64bit
```

For 32-bit applications running on SUSE Linux Enterprise Server 11 on a 64-bit machine, the RPM is:

```
glibc-locale-32bit
```

For 64-bit applications running on SUSE Linux Enterprise Server 11 on a 64-bit machine, the RPM is:

```
glibc-locale
```

Verifying RSCT installation on Linux

To verify that RSCT is installed on a Linux node, enter the following command:

```
rpm -qa | grep -E -e "rsct|src"
```

An output similar to the following example is displayed:

```
src-3.1.5.0-13211
rsct.core.utils-3.1.5.0-13211
rsct.core-3.1.5.0-13211
rsct.basic-3.1.5.0-13211
```

If you installed RSCT components, ensure that they are at a version that applies to this information. In the RPM package names, *platform* is **i386**, **ppc**, **ppc64**, **s390**, or **x86_64**.

Table 11 describes the RSCT for Linux RPM packages.

Table 11. RSCT for Linux packages

This package...	Contains...
rsct.basic-3.1.5.0-13211.*platform*.rpm	• The configuration resource manager • The Group Services subsystem • The Topology Services subsystem

Table 11. RSCT for Linux packages (continued)

This package...	Contains...
rsct.core-3.1.5.0-13211.*platform*.rpm	RSCT core components that includes: • The audit log resource manager • The cluster security services • The event response resource manager (ERRM) • The file system resource manager • The host resource manager • The least-privilege resource manager • The management domain resource manager • The microsensor resource manager • The resource monitoring and control (RMC) subsystem • The sensor resource manager • The system registry • Miscellaneous utilities
rsct.core.cimrm-3.1.5.0-13211.*platform*.rpm	The CIM resource manager (where available)
rsct.core.utils-3.1.5.0-13211.*platform*.rpm	Miscellaneous utilities
rsct.opt.storagerm-3.1.5.0-13211.*platform*.rpm	The storage resource manager
src-3.1.5.0-13211.*platform*.rpm	The system resource controller (SRC)
srclib.64bit-3.1.5.0-13211.*platform*.rpm	The SRC 64-bit libraries
rsct.64bit-3.1.5.0-13211.*platform*.rpm	The RSCT 64-bit library for 64-bit kernel
rsct.opt.stackdump-3.1.5.0-13211.*platform*.rpm	The stack dump function

If the **rpm** command output reveals that the required RSCT RPM packages are not installed, you can install them from the RSCT exploiter's installation media. See the RSCT exploiter's documentation for installation instructions.

You can install RSCT by itself but, because of the dependencies among the RPM packages, the packages must be installed in a specific sequence. In the following instructions, replace *platform* with `i386`, `ppc`, `s390`, `ppc64`, or `x86_64`, as appropriate for your system platform.

Install the RPM packages in the following sequence:

1. Install the SRC by entering the following command:
   `rpm -i src-3.1.5.0-13211.platform.rpm`
2. Install the RSCT utilities by entering the following command:
   `rpm -i rsct.core.utils-3.1.5.0-13211.platform.rpm`
3. Install the RSCT core components by entering the following command:
   `rpm -i rsct.core-3.1.5.0-13211.platform.rpm`
4. Install the RSCT basic components by entering the following command:
   `rpm -i rsct.basic-3.1.5.0-13211.platform.rpm`
5. Optionally, install the RSCT CIM resource manager component by entering the following command:
   `rpm -i rsct.core.cimrm-3.1.5.0-13211.platform.rpm`
6. Optionally, install the RSCT storage resource manager component by entering the following command:

   Note: If you are installing the storage resource manager on a node in a peer domain, take the node offline before you install the storage resource manager component.

   `rpm -i rsct.opt.storagerm-3.1.5.0-13211.platform.rpm`
7. Optionally, install the RSCT stack dump function by entering the following command:

```
rpm -i rsct.opt.stackdump-3.1.5.0-13211.platform.rpm
```

8. Optionally, install the SRC 64-bit libraries by entering the following command:

```
rpm -i srclib.64bit-3.1.5.0-13211.platform.rpm
```

If the **rpm** command output reveals that the previous versions of RSCT RPM packages are installed, you can upgrade RSCT by using the **rpm** command. Before upgrading RSCT, be sure to review the information in "Migration considerations" on page 47.

Do the following steps to upgrade RSCT:

- If your system does not have the **rsct64bit** package installed, you can use the following command to upgrade RSCT:

```
rpm -Fvh src-3.1.5.0-13211.platform.rpm
      rsct.core.utils-3.1.5.0-13211.platform.rpm
      rsct.core-3.1.5.0-13211.platform.rpm
      rsct.basic-3.1.5.0-13211.platform.rpm
     (rsct.core.cimrm-3.1.5.0-13211.platform.rpm
      rsct.opt.storagerm-3.1.5.0-13211.platform.rpm)
```

- If your system has the **rsct64bit** package installed, you can use the following command to upgrade RSCT:

```
rpm -Fvh src-3.1.5.0-13211.platform.rpm
         rsct.core.utils-3.1.5.0-13211.platform.rpm
         rsct.core-3.1.5.0-13211.platform.rpm
         rsct.basic-3.1.5.0-13211.platform.rpm
        (rsct.core.cimrm-3.1.5.0-13211.platform.rpm
         rsct.opt.storagerm-3.1.5.0-13211.platform.rpm)
         rsct.64bit-3.1.5.0-13211.platform.rpm
```

If your system has any RSCT-exploiter packages installed, you might have to upgrade those RPM packages as well. See the RSCT exploiter's documentation for appropriate instructions.

If you want to uninstall RSCT, the packages must be uninstalled in a specific sequence. If any exploiter has a dependency on RSCT, the **rpm** command does not uninstall the RSCT packages.

Uninstall the RPM packages in the following sequence:

1. If the `srclib.64bit` package is installed, uninstall it by entering the following command:

```
rpm -e srclib.64bit
```

2. If the `rsct.64bit` package is installed, uninstall it by entering the following command:

```
rpm -e rsct.64bit
```

3. If the storage resource manager is installed, uninstall it by entering the following command. If the storage resource manager component is on a node in a peer domain, take the node offline before uninstalling the storage resource manager.

```
rpm -e rsct.opt.storagerm
```

4. If the CIM resource manager is installed, uninstall it by entering the following command:

```
rpm -e rsct.core.cimrm
```

5. Uninstall the RSCT basic components by entering the following command:

```
rpm -e rsct.basic
```

6. Uninstall the RSCT core components by entering the following command:

```
rpm -e rsct.core
```

7. Uninstall the RSCT utilities by entering the following command:

```
rpm -e rsct.core.utils
```

8. Uninstall the SRC by entering the following command:

```
rpm -e src
```

The Linux distributions that are supported by this version of RSCT are described in "Supported Linux distributions for RSCT." See your RSCT exploiter's documentation to see whether that particular product also supports a particular distribution.

Supported Linux distributions for RSCT

A number of Linux operating system distributions are supported by RSCT.

Table 12. Supported Linux distributions and hardware platforms

Linux distribution	x86 (System x and BladeCenter®)	AMD64 (System x and BladeCenter)	Intel and AMD processor-based blades (System x and BladeCenter)	POWER4 POWER5 POWER6 POWER7 (Power Systems servers and BladeCenter)	POWER® processor-based blades (Power Systems servers and BladeCenter)	System z®	Power Systems servers
Red Hat Enterprise Linux (RHEL) 5, 6 (6.1, 6.2, 6.3, and 6.4)	yes	yes	yes	yes	yes	yes	yes
SUSE Linux Enterprise Server (SLES) 9, 10, 11 Service Pack 1, 11 Service Pack 2, and 11 Service Pack 3	yes	yes	yes	yes	yes	yes	yes

Related tasks:

"Creating a peer domain" on page 50
Follow these instructions to configure nodes into an RSCT peer domain.

RSCT adapter support for Linux

RSCT for the Linux operating system supports a number of adapters.

RSCT for Linux supports these adapters:

- 10/100 Ethernet adapter.
- 1 GB Ethernet adapter.
- 1 GB Myrinet switch adapter.
- Topspin adapter over InfiniBand. For more information when planning for your network, see "IP addressing on an InfiniBand network" on page 32.
- GX Dual-port 4x InfiniBand Host Channel Adapter (HCA). For more information when planning for your network, see "IP addressing on an InfiniBand network" on page 32.
- Channel bonding configuration support (an arrangement in which two or more network interfaces on a host computer are combined for redundancy or increased throughput).

Migrating to Red Hat Enterprise Linux 5 with RSCT

Migration to Red Hat Enterprise Linux 5 with RSCT involves several considerations.

The first release of RSCT that supports Red Hat Enterprise Linux (RHEL) 5 is RSCT 2.4.6.3. Therefore, when you upgrade a node from RHEL 3 or RHEL 4 to RHEL 5 and RSCT is already installed on that node, if the version of RSCT is not 2.4.6.3, or later, you must either upgrade to RSCT version 2.4.6.3, or later before you upgrade the operating system or upgrade the operating system and RSCT at the same time.

Kernel update requirements

You might need to install one or more kernel updates if you are running RSCT on Linux.

After installing a kernel update, reboot your Linux nodes, especially if you are running Tivoli System Automation, as critical resources might be instantiated on these nodes.

When critical resources are created on a node, error messages similar to the following might appear in **/var/log/messages**:

```
Mar 26 09:12:55 e106e345n04 modprobe: FATAL: Could not load /lib/modules/2.6.5-7
.315-smp/modules.dep: No such file or directory
```

Rebooting the node should fix this problem.

For SUSE Linux Enterprise Server 10 on AMD64 systems:

On AMD64 systems with Linux 10 (SLES 10) installed, RSCT requires Service Pack 1 (SP1) or later.

If SLES 10 SP1 is not installed on an AMD64 system, RSCT functions might fail randomly.

For SUSE Linux Enterprise Server 11:

If you are running RSCT on SLES11 GA, the SLES11 updates are required. Otherwise, RSCT operation might fail due to a problem in the **gettimeofday()** function.

For detailed information about this problem, go to:

http://bugzilla.kernel.org/show_bug.cgi?id=11970

IP addressing on an InfiniBand network

There are issues to consider when configuring IP subnet addressing on a server that is attached to an InfiniBand network on AIX or Linux.

When configuring IP subnet addressing on a server that is attached to an InfiniBand network on AIX or Linux, consider the following:

- *InfiniBand subnets* and *IP subnets* are two distinctly different entities.
- All IP *addresses* in a particular *IP subnet* need to be connected to the same *InfiniBand subnet*.
- Multiple adapters or ports can be connected to the same *InfiniBand subnet*.
- The IP address of each HCA network interface needs to be in a different *IP subnet*. These subnets are used as the basis for RSCT communication groups. RSCT restricts each node so that it only has one local interface in any particular communication group. RSCT does not support the attachment of multiple interfaces on a node to the same *IP subnet*. Therefore, if *n* HCA network interfaces on each node are connected to the same *InfiniBand subnet*, you need to set up *n* separate *IP subnets*.

For more information, see:

- "RSCT adapter support for AIX" on page 26
- "RSCT adapter support for Linux" on page 31

Verifying RSCT installation on Solaris nodes

Use the **pkginfo** command to verify that RSCT is installed on a Solaris node.

To verify that RSCT is installed on a Solaris node, enter the following command:

```
pkginfo | /usr/xpg4/bin/grep -E -e IBM
```

An output similar to the following example is displayed:

```
application IBMrsctBasic          IBM Reliable Scalable Cluster Technology - Basic
application IBMrsctCore           IBM Reliable Scalable Cluster Technology
application IBMrsct-CoreUtils     IBM Reliable Scalable Cluster Technology - Utilities
application IBMrsctOptStoragerm   IBM Reliable Scalable Cluster Technology - StorageRM
application IBMsrc                IBM System Resource Controller
application IBMrsct64bit          IBM Reliable Scalable Cluster Technology - 64-bit
```

Table 13 on page 33 describes the RSCT for Solaris packages.

Table 13. RSCT for Solaris RPMs

This Solaris package...	Contains...
IBMsrc-3.1.5.0-13211.sparc.pkg	The system resource controller (SRC)
IBMsrclib64bit-3.1.5.0-13211.sparc.pkg	SRC 64-bit libraries
IBMrsctCoreUtils-3.1.5.0-13211.sparc.pkg	Miscellaneous utilities
IBMrsctCore-3.1.5.0-13211.sparc.pkg	RSCT core components that includes: • The audit log resource manager • The cluster security services • The event response resource manager (ERRM) • The host resource manager • The least-privilege resource manager • The resource monitoring and control (RMC) subsystem • The system registry
IBMrsctBasic-3.1.5.0-13211.sparc.pkg	• The configuration resource manager • The Group Services subsystem • The Topology Services subsystem
IBMrsctOptStoragerm-3.1.5.0-13211.sparc.pkg	The storage resource manager
IBMrsct64bit-3.1.5.0-13211.sparc.pkg	The RSCT 64-bit library for the 64-bit kernel

If the **pkginfo** command output reveals that the required RSCT packages are not installed, you can install them from the RSCT exploiter's installation media. See the RSCT exploiter's documentation for installation instructions.

You can install RSCT by itself but because of the dependencies among the packages, the packages must be installed in a specific sequence.

Install the packages in the following sequence:

1. Install the system resource controller by entering the following command:
   ```
   pkgadd -d  IBMsrc-3.1.5.0-13211.sparc.pkg all
   ```
2. Install the RSCT utilities by entering the following command:
   ```
   pkgadd -d IBMrsctCoreUtils-3.1.5.0-13211.sparc.pkg all
   ```
3. Install the RSCT core components by entering the following command:
   ```
   pkgadd -d IBMrsctCore-3.1.5.0-13211.sparc pkg all
   ```
4. Install the RSCT basic components by entering the following command:
   ```
   pkgadd -d IBMrsctBasic-3.1.5.0-13211.sparc pkg all
   ```
5. Optionally, install the RSCT storage resource manager component by entering the following command. If you are installing the storage resource manager on a node in a peer domain, take the node offline before you install the storage resource manager component.
   ```
   pkgadd -d IBMrsctOptStoragerm-3.1.5.0-13211.sparc.pkg all
   ```
6. Optionally, install the RSCT 64-bit package to run 64-bit applications by entering the following command:
   ```
   pkgadd -d IBMrsct64bit-3.1.5.0-13211.sparc.pkg all
   ```
7. Optionally, install the SRC 64-bit package to run SRC 64-bit client applications by entering the following command:
   ```
   pkgadd -d IBMsrclib64bit-3.1.5.0-13211.sparc.pkg all
   ```

If you want to uninstall RSCT, the packages must be uninstalled in a specific sequence. If any exploiter has a dependency on RSCT, the **pkgrm** command does not uninstall the RSCT packages. Uninstall the packages in the following sequence:

1. If the SRC 64-bit package is installed, uninstall it by entering the following command:
   ```
   pkgrm IBMsrclib64bit
   ```
2. If the RSCT 64-bit package is installed, uninstall it by entering the following command:
   ```
   pkgrm IBMrsct64bit
   ```
3. If the storage resource manager is installed and is on a node in a peer domain, take the node offline before you uninstall the storage resource manager. To uninstall the storage resource manager component, enter the following command:
   ```
   pkgrm IBMrsctOptStoragerm
   ```
4. To uninstall the RSCT basic components, enter the following command:
   ```
   pkgrm IBMrsctBasic
   ```
5. To uninstall the RSCT core components, enter the following command:
   ```
   pkgrm IBMrsctCore
   ```
6. To uninstall the RSCT utilities, enter the following command:
   ```
   pkgrm IBMrsctCoreUtils
   ```
7. To uninstall the system resource controller, enter the following command:
   ```
   pkgrm IBMsrc
   ```

Related tasks:

"Creating a peer domain" on page 50
Follow these instructions to configure nodes into an RSCT peer domain.

Update requirement for Solaris 10

RSCT requires a Solaris 10 update.

To function properly, RSCT requires Solaris 10 update 1 (Update 01/06) or later.

Verifying RSCT installation on the Windows platform

RSCT is automatically installed when you install Tivoli System Automation on a Windows system.

After installing Tivoli System Automation, complete the following steps to verify that RSCT will run successfully:

1. Check that the principal user SYSTEM has at least read access to the file *C:\WINDOWS\system32\drivers\etc\services*. If you are using the local Administrator user with Tivoli System Automation, this user must have read and write access to the file. In general, you must grant the local administrator or Windows domain administrator full control of the file.

   You can either use Windows Explorer to view and change the access permissions or use the Windows **cacls** command. The output of the **cacls** command will look like this:

   ```
   > cacls services
   C:\WINDOWS\system32\drivers\etc\services NODE2\Administrator:F
                                           NT AUTHORITY\SYSTEM:R
   ```
2. Verify that **/etc/services** is symbolically linked to **/dev/fs/C/WINDOWS/system32/drivers/etc/services**.
3. Enable system logging in a Windows environment. By default, the system logger (**syslogd**) is not enabled in a Subsystem for Unix-based Applications (SUA) environment. To enable logging, edit the **/etc/init.d/syslog** file and uncomment the statements that automatically start the **syslogd** daemon.

   The following is an excerpt of the statements in **/etc/init.d/syslog** that need to be uncommented:

   ```
   ....
                  ${SYSLOGD}
                  [ $? = 0 ] && echo "syslogd started"
                  ;;
   .....
   ```

   Syslog messages will be stored in the **/var/adm/log/messages** directory.
4. Verify and, if needed, enable SRC by issuing the following commands:

```
ps -efX unix
/sbin/srcmstr &
```

5. Verify the existence of RSCT's cluster authentication services and RMC subsystems and, if needed, start the RMC subsystem. To do so, run the following command to list the RMC (**ctrmc**) and the cluster authentication services (**ctcas**) subsystems:

   ```
   /usr/bin/lssrc -a
   ```

   The output should resemble the following:

   ```
   Subsystem         Group            PID     Status
   ctrmc             rsct             27085   active
   ctcas             rsct                     inoperative
   ```

   If the RMC subsystem (**ctrmc**) is not active (that is, its status shows that it is inoperative), manually start it by issuing the following command:

   ```
   startsrc -s ctrmc
   ```

   The status of the cluster authentication services subsystem should show that it is inoperative. Do not start the cluster authentication services subsystem.

6. Verify that **/usr/bin** is linked to **/bin**. For example:

   ```
   $ ls -l /usr/bin
   lr--r--r--  1 Administrator  +Administrators  4 Jun 28 11:27 /usr/bin -> /bin
   ```

7. Verify that **/lib>** is linked to **/usr/lib**. For example:

   ```
   $ ls -l /lib
   lr--r--r--  1 Administrator  None  8 Jun 28 11:26 /lib -> /usr/lib
   ```

Related tasks:

"Creating a peer domain" on page 50
Follow these instructions to configure nodes into an RSCT peer domain.

Administering an RSCT peer domain

To achieve high availability, you can configure a cluster of nodes into an RSCT peer domain.

Using configuration resource manager commands, you can:

- Create a peer domain
- Add nodes to an existing peer domain
- Take a peer domain node or an entire peer domain offline
- Remove a node from a peer domain or remove an entire peer domain
- List and modify various aspects of the peer domain configuration

Related concepts:

"Resource managers provided with RSCT" on page 110
Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

"Resource classes defined by the configuration resource manager" on page 114
In general, you do not need to manipulate resources of these classes directly. Instead, use the configuration resource manager commands.

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

An RSCT peer domain

An *RSCT peer domain* is a cluster of nodes configured for high availability.

The peer domain could consist of all nodes in your cluster, or could be a subset of nodes in your overall cluster solution (which could also consist of nodes configured into a management domain). Within a management domain, however, the management server cannot belong to the same peer domain as any of the managed nodes.

An RSCT peer domain uses:

- RSCT cluster security services for authentication.
- The Topology Services subsystem for node/network failure detection. Generally, the peer domain's use of this subsystem is transparent to you.
- The Group Services subsystem for cross node/process coordination. Generally, the peer domain's use of this subsystem is transparent to you.
- The resource monitoring and control subsystem for coordination between the various RSCT subsystems. Generally, the peer domain's use of this subsystem is transparent to you. However, you can use RMC to monitor the peer domain.

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

"The Topology Services subsystem" on page 286
In an RSCT peer domain, the configuration resource manager uses the Topology Services subsystem to monitor the liveness of the adapters and networks included in communication groups.

"The Group Services subsystem" on page 308
The configuration resource manager uses the Group Services subsystem to provide distributed coordination, messaging, and synchronization among nodes in an RSCT peer domain.

"Monitoring resources using RMC and resource managers" on page 126
The Resource Monitoring and Control (RMC) subsystem is the scalable backbone of RSCT that provides a generalized framework for managing and monitoring resources (physical or logical system entities) within a single system or a cluster.

Configuration resource manager

The configuration resource manager provides the ability to create and administer an RSCT peer domain.

This is essentially a management application implemented as an RMC resource manager. A command-line interface to this resource manager enables you to create a new peer domain, add nodes to the domain, list nodes in the domain, and so on.

Communication groups

Communication groups control how liveness checks (in other words, Topology Services' heartbeats) are performed between the communication resources within the peer domain. Each communication group corresponds to a Topology Services' heartbeat ring, and identifies the attributes that control the liveness checks between the set of network interfaces and other devices in the group.

The configuration resource manager automatically forms communication groups when a new peer domain is formed. When you then bring a peer domain online, the configuration resource manager will supply the communication group definition to Topology Services. Topology Services will create the actual heartbeat rings needed to perform liveness checks for the peer domain nodes.

Each communication group has several characteristics. These characteristics specify:

- the number of missed heartbeats that constitute a failure
- the number of seconds between the heartbeats
- whether or not broadcast should be used
- whether or not source routing should be used

Each communication group also has a list of its member network interfaces.

The configuration resource manager may also form new communication groups as new nodes are added to the peer domain. When these added nodes are brought online in the peer domain, the configuration resource manager supplies the modified information to Topology Services. Topology Services may then modify existing heartbeat rings or create additional heartbeat rings.

In general, communication groups will be transparent to you. The configuration resource manager forms them in conjunction with the Topology Services subsystem as you issue commands to create and modify a peer domain. Although the configuration resource manager allows you to create your own communication groups, such manual configuration is neither necessary or advisable.

Quorum

Quorum refers to the minimum numbers of quorum nodes within the peer domain that are required to carry out a particular operation.

There are three kinds of quorum that specify the number of nodes required for different types of operations. These are *startup quorum*, *configuration quorum*, and *operational quorum*.

Quorum node:

When changing cluster configuration data or global resource manager replicated global data, RSCT guarantees subsequent discovery of the latest changes by not permitting them unless an *N*/**2 + 1** quorum of all defined nodes participate in committing the changes. The use of *quorum nodes* reduces this number of required online nodes by applying the *N*/**2 + 1** formula only to a subset of pre-defined nodes, that is, quorum nodes.

Quorum nodes are distinguished from non-quorum nodes by the value of an **IsQuorumNode** attribute in the PeerNode class. The basis for determining whether a quorum of quorum nodes exists must remain stable for the duration of configuration data updates, global resource manager data updates, or both. This requires coordination between the mechanism for changing quorum node configuration (in the configuration resource manager) and global resource managers. This is addressed by a function of the configuration resource manager and the resource manager framework called *quorum coordination,* through which the configuration resource manager asks all global resource managers whether they support a quorum set change, and performs the change only if they all reply in the affirmative.

When creating a peer domain, you can use the **mkrpdomain** command's **-f** option with a node definition file to specify which nodes are to be quorum nodes and whether they have access to the peer domain's tiebreaker mechanism. In the node definition file, node names that are followed by **@Q** will be considered as quorum nodes. Node names that are followed by **@!Q** will be considered as non-quorum nodes. For details about how to specify **@Q** and **@!Q** in a node definition file, see the description of the **-f** option under the **mkrpdomain** command in the *Technical Reference: RSCT for AIX* or the *Technical Reference: RSCT for Multiplatforms*.

When used in the context of quorum, discussions of *nodes* in this document refer to *quorum nodes.*

Quorumless domain:

A peer domain can be configured to operate without quorum requirements. When normal (strong) quorum is disabled, the **IsQuorumNode** attribute is not used. The number of nodes that is required to satisfy the startup, configuration, and operational quorum is always one. However, if domain partitioning happens, a tiebreaker is always used to resolve operational quorum. When operating without quorum restrictions, configuration changes can be made independently to domain partitions. When the partitions rejoin, the configuration changes that are made to one partition are overwritten by the other partition.

During the start of a peer domain that is configured to be quorumless, the configuration resource manager tries to contact all of the cluster nodes to discover the configuration that contains the most recent changes. The **QuorumLessStartupTimeout** attribute of the **IBM.PeerNode** resource class

determines the maximum time that is spent in attempting to contact the remote nodes while bringing the domain online. After all of the nodes are contacted or the timeout expires, the domain comes online with the most recently determined configuration among all of the reachable nodes. Nodes that join the domain later conform to the configuration of the online domain. Although the configuration resource manager attempts to locate the most recent peer domain configuration, there is no assurance which configuration might be chosen if changes are made while the cluster is partitioned. The configuration can be reverted if a node with a later version is inactive or unreachable.

Likewise, the configuration of global resource managers that are operating in a quorumless peer domain can be reverted when a node or domain is brought online. Global resource managers that are running in a quorumless domain come operational with the configuration that is associated with the first node to become online. This type of quorum is therefore only recommended for specific environments where the configuration data is static or can be rediscovered by the resource managers; for example, PowerHA or the shared storage pool in Cluster-Aware AIX (CAA) domains.

Startup quorum:

Startup quorum refers to the number of nodes needed to bring a peer domain online. If the configuration resource manager is unable to reach this minimum number of nodes, it will not be able to start the peer domain.

Configuration quorum:

Configuration quorum refers to the minimum number of nodes, or a certain peer-domain state, needed to perform operations that modify the peer domain's configuration information. If you issue a command that will modify a peer domain's configuration, and the configuration resource manager is unable to reach this minimum number of nodes, the command will fail.

Operational quorum:

Operational quorum refers to the minimum number of nodes, or a certain peer-domain state, needed to safely activate resources without creating conflicts with another subdomain. It is used to protect data following domain partitioning.

About domain partitioning:

Domain partitioning refers to the inadvertent division of a peer domain into two or more subdomains.

How operational quorum helps the configuration resource manager protect data following domain partitioning:

Following domain partitioning, when critical resources are active on nodes, the configuration resource manager needs to determine which subdomain can continue operating and which other(s) should be dissolved.

This is especially important when there are applications running on the domain which use shared resources. If the peer domain is partitioned, nodes in one subdomain are no longer aware of nodes in any other subdomain. Data corruption can occur if nodes in different sub-domains try to access the same shared resource. The configuration resource manager prevents this situation by deciding which subdomain has operational quorum and can continue operating, thus becoming the peer domain. Usually, the subdomain with the majority of nodes will have operational quorum.

How the configuration resource manager uses a tiebreaker to determine operational quorum:

After domain partitioning, it is usually the subdomain with the majority of nodes that has operational quorum. However, sometimes there is a tie in which multiple subdomains have exactly half of the defined nodes. A tie situation also occurs when exactly half the nodes of a domain are online and the other half are inaccessible.

When there is a tie, the configuration resource manager uses a tiebreaker to determine which subdomain has operational quorum. A tiebreaker is a resource defined by the configuration resource manager that specifies how tie situations must be resolved. It is the tiebreaker that determines which subdomain has operational quorum, and so survives, and which subdomain is dissolved.

In **QuorumLess** quorum type mode, after domain partitioning, the configuration resource manager always uses a tiebreaker to determine the subdomain that has an operational quorum.

How the configuration resource manager uses critical resource protection methods:

When a subdomain that has critical resources loses quorum, the configuration resource manager uses a *critical resource protection method* on each node of the subdomain to ensure that critical resources will not be corrupted. A *critical resource protection method* is software that determines how the configuration resource manager will respond when quorum is lost in a subdomain.

A critical resource protection method will also be used on a node whose configuration resource manager, Group Services, or Topology Services daemon hangs. The configuration resource manager defines a number of critical resource protection methods. You can specify a critical resource protection method for use by an entire peer domain or one particular node. Critical resource protection methods do such things as halt the system, reset and reboot the system, and so on.

Quorum types:

The quorum type of a peer domain specifies how startup quorum, configuration quorum, and operational quorum are calculated for the peer domain.

The quorum types are as follows:

Normal
Normal mode is the default for an AIX or Linux cluster. In this mode, the startup quorum, configuration quorum, and operational quorum are calculated by the following method:

StartupQuorum = N/2
ConfigQuorum = N/2 + 1
OpQuorum = Majority + tiebreaker

Quick Quick startup mode is useful for large clusters. In this mode, the startup quorum, configuration quorum, and operational quorum are calculated by the following method:

StartupQuorum = 1
ConfigQuorum = N/2 + 1
OpQuorum = Majority + tiebreaker

Override
Override mode, available only for IBM i environments, is the default for such environments. In this mode, the startup quorum, configuration quorum, and operational quorum are calculated by the following method:

StartupQuorum = 1
ConfigQuorum = 1
OpQuorum is externally provided by RMC exploiter.

SANFS

SANFS mode, available only for environments with the IBM TotalStorage SAN File System, is the default for such environments. In this mode, the startup quorum, configuration quorum, and operational quorum are calculated by the following method:

StartupQuorum = 1
ConfigQuorum is externally provided by a designated group state value.
OpQuorum = Majority + tiebreaker

QuorumLess

QuorumLess mode suspends quorum rules so that the domain startup operation and changes are allowed for any number of nodes. In this mode, the startup quorum, configuration quorum, and operational quorum are calculated by the following method:

StartupQuorum = 1
ConfigQuorum = 1
OpQuorum = 1
OpQuorum = 1 + tiebreaker (for domain partitions)

The quorum type of a domain can be changed with the **chrsrc** command by entering the following command:

```
> chrsrc –c IBM.PeerNode QuorumType=quorum_type
```

A majority of domain nodes are required to be online when you change the quorum type from **QuorumLess** to **Normal** (strong).

Peer domain on Cluster Aware AIX

Cluster Aware AIX (CAA) introduces fundamental clustering capability to AIX, including definition of the set of nodes that comprise the cluster, and detection of node and interface liveness.

When RSCT operates on nodes in a CAA cluster, a peer domain is created that is equivalent to and represents the CAA cluster, and can be used to manage the cluster by using peer domain commands and interfaces. This peer domain presents largely the same set of function to users and software as other peer domains not based on CAA. But where a peer domain, which is operating without CAA, autonomously manages and monitors the configuration and liveness of the nodes and interfaces that it comprises, the peer domain that represents a CAA cluster acquires configuration information and liveness results from CAA. It introduces some differences in the mechanics of peer domain operations, but very few in the view of the peer domain that is exposed to the users.

Only one CAA cluster can be defined on a set of nodes. Therefore, if a CAA cluster is defined then the peer domain that represents it is the only peer domain which can exist, and it exists and be online for the life of the CAA cluster. If no CAA cluster is configured, then existing and new peer domains that operate in the autonomous mode can be used with no change in function. CAA function is only available on AIX, and on all other supported platforms, peer domain function operating in the autonomous mode is unchanged.

A CAA cluster and the equivalent RSCT peer domain operate hand in hand such that a change made to the CAA cluster by using CAA commands and interfaces, is reflected automatically in the corresponding peer domain; similarly the existing peer domain commands applied to a peer domain operating on CAA result in equivalent changes to the CAA cluster. So, for example, when you create a CAA cluster by using **mkcluster** command, the equivalent peer domain also gets created as a result, while the **mkrpdomain** command can be used to create a peer domain and the underlying CAA cluster in one operation. Similarly node add and delete operations that use either peer domain or cluster commands are applied to both the CAA cluster and the peer domain.

Because a peer domain operating on CAA exists and is online for the life of the CAA cluster, some operations that are relevant to peer domains not operating on CAA, **startrpdomain**, **stoprpdomain**, **startrpnode**, and **stoprpnode**, have no effect on a peer domain that is running on a CAA cluster. Also, the **preprpnode** command, which is used to exchange public keys among nodes in preparation for peer domain creation, is not needed when a peer domain based on a CAA cluster is to be created.

In a peer domain that represents a CAA cluster, the definition of quorum might change depending on the installed AIX release.

RSCT version 3.1.2.0, or later, can be installed on the nodes and can coexist with prior RSCT releases. Until RSCT is upgraded to at least version 3.1.2.0 on all of the cluster nodes, the quorum is determined by the ability of the group leader node of the configuration resource manager to connect to the CAA services and acquire configuration information. If the group leader node loses this ability, it releases the group leader role to some other node in the cluster.

After the RSCT level is upgraded to version 3.1.2.0, or later, on all of the cluster nodes and for all domains that represent CAA clusters on AIX 6.1 with Technology Level 7 or AIX 7.1 with Technology Level 1, quorum is determined by the value of the **QuorumType** attribute of the **IBM.PeerNode** resource class.

For all of the RSCT release levels on CAA clusters, peer domain operations can be used to change the configuration of the peer domain and the CAA cluster. CAA services provide RSCT with the authorized view of the cluster node configuration and activities. The quorum definition that it uses determines when configuration can be changed by using RSCT operations. The changes can affect either global resource configuration that is specific to RSCT or CAA cluster configuration.

Peer domain on CAA linked clusters

Cluster Aware AIX (CAA) on IBM AIX 6 with Technology Level 8 and IBM AIX 7 with Technology Level 2 introduces the concept of linked clusters to provide multisite clustering solutions. CAA clusters defined at each site communicate across established network links to form a linked cluster. An RSCT peer domain that operates on linked clusters encompasses all nodes at each site. That is, the nodes that comprise each site cluster are all members of the same peer domain.

When the site links are broken, the clusters that define each site operate independently and might diverge when fragmented. Peer domain services can be configured to protect critical cluster resources when site splits occur.

When the network links are repaired, the diverged sites are merged again to form a unified linked cluster. This merge operation synchronizes the independent CAA clusters and joins the peer domain partitions. During the merge operation, the repository differences of CAA clusters between the sites are reconciled by replicating from one side of the site partition to the other. Cluster services are also restarted on a site to rejoin the site and peer domain partitions.

The site whose cluster configuration prevails and remains operational is the dominant site. The configuration of the peer domain and CAA clusters decide the dominant site and the actions to be taken during split and merge operations.

Site merge

CAA defines policies that determine the dominant site during a merge operation. The merge policy is configured by using the following CAA command:

/usr/sbin/clctrl -tune -o site_merge_policy = [h] | m | p]

The site merge policies are described in the following table:

Table 14. Site merge policy

Flag	Site merge policy name	Description
h	Heuristic	The dominant site during a merge operation is determined by the configuration of the RSCT peer domain that is operating on the CAA linked clusters. Site reconciliation is automatic or manual depending on the quorum type and tiebreaker properties of the peer domain configuration.
m	Manual	No automatic action is taken during the site merge operation. The administrator must manually reconcile the site by running the commands to synchronize and rejoin the linked clusters. See Cluster Management for information about reconciling a manual site merge operation.
p	Priority	A numeric priority value is assigned to each site when the CAA site clusters are defined. During a site merge operation, the site with the lowest priority (highest priority value) is synchronized to the highest priority site (lowest priority value).

When linked sites merge, the peer domain partitions that are running on each site are rejoined by recycling RSCT on the losing site. Similarly, the cluster configurations of a CAA site are merged by reconciling configuration differences to match the dominant site as determined by the policy. For priority and heuristic policies, the CAA site resynchronizes and reboots the nodes on the losing site to complete the merge operation automatically. If the site merge policy is manual, the sites are rejoined by using commands.

Site split

When sites split, each side might continue to operate independently. Also, the peer domain that is running on the linked clusters creates partitions into two sides. One side of the peer domain partition is granted operational quorum as determined by the quorum type and tiebreaker configuration of the peer domain. To protect cluster resources, the critical protection services of the peer domain can be configured to act when a site splits, such as halting or rebooting nodes on the losing side of the sunder. Critical resources are protected on all active nodes in the site that does not acquire or retain operational quorum after the site splits.

CriticalMode attribute

The persistent attribute of the **IBM.PeerNode** class, **CriticalMode**, is an enhancement to critical protection in case of peer domain partition. The **CriticalMode** attribute can be set such that all of the cluster nodes must be considered as having active critical resources. This setting can be useful for multisite configurations to run the critical protection method on all the nodes of the losing side of a split site.

Subsite split and merge

Network failures within the individual sites can result in local peer domain partitions when linked sites are split. During local site partition, the peer domain acts according to the operational quorum and according to the critical protection method configuration of the cluster. When the local site partitions merge, CAA services are not required to synchronize when linked sites are reconciled.

PowerHA site and stretched clusters

A PowerHA stretched cluster or a cluster that is running on multisite CAA clusters can be configured with the following values of the **IBM.PeerNode** class attributes:

Table 15. Configuration of IBM.PeerNode class attributes for PowerHA stretched clusters

Attributes and values	Description
QuorumType = 4 (QuorumLess)	The peer domain operates without requiring a majority of the quorum. It allows local site clusters to operate independently when they are not linked.
OpQuorumTieBreaker = Operator | SCSIPR | Success	The **Operator** or an automatic tiebreaker, such as **SCSIPR**, can be used so that the critical protection method is run when sites split or when subsite domains are partitioned. The tiebreaker is also used to determine the dominant site during a site merge operation for the heuristic policy. If critical resource protection is not required, the **Success** tiebreaker can be used so that partitions remain operational following site and domain sunders. If the site merge policy is heuristic and the **Success** tiebreaker is used, the dominant site during the merge operation is the partition with the larger number of joined members.
CriticalMode = 2	The **CriticalMode** attribute can be enabled so that the critical resource is protected on the losing side of the split site and subsite partitions.
NotifyQuorumChangedCommand = Yes	PowerHA SystemMirror® can configure this attribute to receive notification that manual intervention is required to resolve site and domain split or merge operations when the **Operator** tiebreaker is used.

Related information:

Cluster management

clctrl command

RSCT compatibility on CAA

RSCT 3.1.2.0, or later can be installed on the nodes and can coexist with the peer domains that represent existing CAA clusters. However, RSCT releases before version 3.1.2.0 cannot coexist and cannot be used to create CAA clusters on AIX 6.1 with Technology Level 7 or AIX 7.1 with Technology Level 1. When you create a peer domain that represents a CAA cluster on these newer AIX levels, all of the nodes must be installed with RSCT 3.1.2.0, or later.

Apart from such differences, the configuration resource manager function available in peer domains not operating on CAA is available unchanged in a peer domain based on a CAA cluster.

Because CAA delivers fundamental node and interface liveness information, the Topology Services subsystem is not active in a peer domain based on CAA. The remaining subsystem complement active in a peer domain not operating on CAA, is active in a peer domain based on CAA as well.

When creating CAA linked clusters, RSCT 3.1.4.0 or later must be installed on all nodes.

Configuration resource manager commands

Configuration resource manager commands are useful when performing a range of tasks.

Table 16 outlines the tasks you can perform using configuration resource manager commands.

Table 16. Configuration resource manager commands and their tasks

To perform this task...	Use these configuration resource manager commands...	For more information, see:
Create a peer domain	1. **preprpnode** to prepare the security environment on each node that will participate in the peer domain. 2. **mkrpdomain** to create a new peer domain definition. 3. **startrpdomain** to bring the peer domain online.	"Creating a peer domain" on page 50
Add nodes to an existing peer domain	1. **preprpnode** to prepare the security environment on the new node. 2. **addrpnode** to add the node to a peer domain. 3. **startrpnode** to bring the node online.	"Adding nodes to an existing peer domain" on page 60
Take a peer domain node offline	**stoprpnode**	"Taking a peer domain node offline" on page 64

Table 16. Configuration resource manager commands and their tasks (continued)

To perform this task...	Use these configuration resource manager commands...	For more information, see:
Take a peer domain offline	**stoprpdomain**	"Taking a peer domain offline" on page 65
Remove a node from a peer domain	**rmrpnode**	"Removing a node from a peer domain" on page 66
Remove a peer domain	**rmrpdomain**	"Removing a peer domain" on page 66
List communication groups	**lscomg**	"Listing communication groups" on page 71
Modify a communication group's characteristics (Topology Services' tunables)	**chcomg** to: • Specify the communication group's sensitivity setting (the number of missed heartbeats that constitute a failure). • Specify the communication group's period setting (the number of seconds between heartbeats). • Specify the communication group's priority setting (the importance of this communication group with respect to others). • Specify the communication group's broadcast setting (whether or not to broadcast if the underlying network supports it). • Specify the communication group's source routing setting (in case of adapter failure, whether or not source routing should be used if the underlying network supports it).	"Modifying a communication group's characteristics" on page 73
Manually configure communication groups *(not necessary under normal circumstances; only to be exercised in unavoidable situations)*	**chcomg** to modify a communication group's network interface	"Modifying a communication group's network interface" on page 75
	mkcomg to create a communication group	"Creating a communication group" on page 77
	rmcomg to remove a communication group	"Removing a communication group" on page 78
Make IPv6 addresses visible in the **IBM.NetworkInterface** resource class and use them for heartbeating and internal peer domain operations.	Use the **mkrpdomain** command **-6** flag.	"Creating a peer domain definition" on page 52

In addition to the tasks you can perform by using configuration resource manager commands, you can also use generic RMC commands to:

- Modify Topology Services and Group Services parameters.
- Determine how the configuration manager responds to domain partitioning to prevent corruption of critical data.

When describing how to perform these administrative tasks, the command examples do not necessarily contain descriptions of each command option. For complete information about any of the commands, see the *Technical Reference: RSCT for AIX* or the *Technical Reference: RSCT for Multiplatforms*. If you encounter error messages while trying to perform the tasks, see the *Messages for RSCT* for recovery information.

Prerequisites and restrictions when using configuration resource manager commands

There are various prerequisites and restrictions when using configuration resource manager commands.

Before using configuration resource manager commands, be aware of the following prerequisites and restrictions:

- The following packages are required:

- rsct.basic
- rsct.core
- rsct.core.sec (required for AIX nodes only)
- rsct.core.utils

  On AIX, these packages are available as part of the base AIX operating system. On other platforms, these packages are shipped with the products (such as PowerHA SystemMirror and TSA) that use the RSCT technology, and must have been installed as part of the product installation procedure.

- All of the nodes that you plan to include in the peer domain must be reachable from all other nodes. While you can have multiple networks and routers to accomplish this, there must be IP connectivity between all nodes of the peer domain.
- All of the network interfaces that you plan to include in the peer domain, and that are in the same subnet, must be configured to use the same maximum transmission unit (MTU) size. It is important because MTU size differences between interfaces in the same subnet can result in packet loss.
- In a Cluster-Aware AIX (CAA) environment, some of the configuration resource manager commands might not be supported. Refer to each of the command entries in the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* for details.
- Any configuration resource manager commands that are supported in a CAA environment impacts the AIX cluster as well as the RSCT layer.
- Avoid running configuration resource manager commands on a node that supports PowerHA SystemMirror 7.1, or later. Any cluster operations or changes must be accomplished through the PowerHA SystemMirror product by using its supported commands.

Addressing resource contention

On heavily-loaded systems, contention for resources such as memory, I/O, or CPU could result in RSCT daemons not being able to make progress in a timely manner. This contention could result in false node failures or in RSCT daemons being recycled.

On AIX and Linux systems, the Topology Services, Group Services, and configuration resource manager daemons run with a fixed realtime CPU priority, to minimize the possibility that these daemons are prevented from accessing system resources. Fixed realtime CPU priority allows these daemons to access CPU resources even when several other processes in the system are running. The use of a realtime fixed CPU priority does not result in the RSCT daemons using additional CPU resources. This priority only ensures that the daemons are allowed to access the CPU whenever needed.

The second step in improving the daemons' resilience to resource contention involves locking (or "pinning") their pages in real memory. Once the pages are brought into physical memory, they are not allowed to be paged out. This minimizes the possibility of daemons becoming blocked or delayed during periods of high paging activity.

Because the daemons' pages are locked in memory, the corresponding physical pages are dedicated to the daemons and cannot be used by other processes in the system. Therefore, the amount of physical memory available for other processes is slightly reduced.

By default, the daemons use a fixed CPU priority and lock the pages in memory. This behavior can be changed, however, with the use of the **cthatstune** command.

The following command directs the RSCT daemons to not use a fixed CPU priority:

```
/usr/sbin/rsct/bin/cthatstune -p 0
```

For the Group Services daemon, the setting only takes effect the next time the RSCT peer domain is online on the node.

The following command directs the RSCT daemons to not lock their pages in memory:

```
/usr/sbin/rsct/bin/cthatstune -m NONE
```

The setting only takes effect the next time the RSCT peer domain is online on the node.

The following command has the same effect:

```
CT_MANAGEMENT_SCOPE=2 chrsrc -c IBM.RSCTParameters \ TSPinnedRegions=256
```

Supported RSCT versions

RSCT peer domains are supported in RSCT 2.2.1.20, or later. Although it is possible to create an RSCT peer domain with an earlier version (RSCT 2.2.1.10), that version is not supported. Nodes running RSCT 2.2.1.10 should *not* be added to an RSCT peer domain that was created with RSCT 2.2.1.20, or later.

To verify that the RSCT version was installed, enter:

```
/usr/sbin/rsct/install/bin/ctversion
```

For earlier versions of RSCT that might not support the **ctversion** command, use the following alternative instructions.

To verify that the RSCT version was installed on an AIX node, enter:

```
lslpp -l rsct*
```

To verify that the RSCT version installed on a Linux node, enter the command:

```
rpm -qa | grep rsct
```

The components of a given version of RSCT and, in particular, the configuration resource manager and its peer domain management functionality, are aware of all prior RSCT versions and are able to correctly compare their current version to all preceding ones, taking into account instances of differently numbered versions delivering the same level of functionality. However, the reverse is not true—that is, earlier versions of RSCT are not necessarily able to correctly compare the version identifiers of newer versions of RSCT.

When you create a peer domain with a set of nodes that have different versions of RSCT installed on them, run the **mkrpdomain** command on the node with the latest (that is, highest) version of RSCT. If you run **mkrpdomain** on a node with an older version of RSCT, that node might not correctly evaluate the version identifiers of newer versions installed on other nodes and, consequently, the peer domain will not function correctly.

The same is true when you add nodes to an existing peer domain whose nodes already have mixed versions of RSCT installed on them. (That is, the value of the **MixedVersions** attribute of the **IBM.PeerDomain** class is **yes**.) Run the **addrpnode** command on a node in the peer domain that has the latest version of RSCT, as version comparisons for the candidate nodes will be done on this node prior to adding the candidates.

Even when following this practice, it is possible that certain version combinations might still cause errors, particularly in cases that involve nodes with an installed RSCT version prior to 2.4.1. If it occurs, contact IBM support for assistance with your particular situation.

Cluster-Aware AIX is supported with RSCT 3.1.0.0, and later.

Support for virtual IP address (VIPA) interfaces

As of RSCT releases 2.3.5 and 2.4.1, the AIX implementation of RSCT allows you to configure virtual IP address (VIPA) interfaces on nodes that are part of a peer domain or a management domain.

Once VIPA is configured to include a set of physical network interfaces, outgoing packets that go through one of these interfaces will have the VIPA interface address as source address. As a result, the destination of the packet, and intervening routers, needs to have a route that allows it to communicate back with the VIPA. These routes are needed even if there is no application on the nodes that communicates using the VIPA. Failure to create these routes may result in nodes that fail to communicate through TCP or UDP, even though the **ping** command may still show connectivity to be intact.

Unlike other network interfaces, VIPA interfaces are not monitored using Topology Services' heartbeating. The output of the **lsrsrc IBM.NetworkInterface** command will reveal that the **HeartbeatActive** attribute is set to **0** for all VIPA interfaces. Manually attempting to set the attribute to **1** will have no effect.

For details on how to configure VIPA interfaces, see *AIX System Management Guide: Communications and Networks*, SC23-4909.

For information on whether VIPA is supported by a particular RSCT exploiter, see the exploiter's product documentation. Special attention is required regarding which of the interfaces on a node are allowed to be part of the VIPA and a particular RSCT exploiter might require that given networks should not be made part of VIPA.

Migrating a peer domain to a new version of RSCT

There are several considerations to be aware of before you migrate an RSCT peer domain and update the active RSCT version to a new level.

Migration considerations

Review the following considerations before you migrate one or more nodes of an RSCT peer domain to a newer level.

Before you migrate one or more nodes of an RSCT peer domain to a newer level, review the following considerations to determine if any of them are applicable:

Avoid domain partitioning when you migrate from RSCT 2.2.1.*n* or 2.3.0.*n*

AIX 5.1 nodes that run with the RSCT level 2.2.1.*n* and AIX 5.2 nodes that run with the RSCT level 2.3.0.*n* cannot be migrated to RSCT version 2.3.3.0, or later while online in a peer domain that contains nodes that run with a level of RSCT 2.3.1.*n*, or later. If nodes that run RSCT 2.2.1.*n* or 2.3.0.*n* are migrated while online in a peer domain that contains RSCT 2.3.1.*n*, or later nodes, a partitioned peer domain can be created when the migration completes.

Before you migrate an individual node that runs RSCT 2.2.1.*n* or 2.3.0.*n*, take the node offline using the **stoprpnode** command (as described in "Taking a peer domain node offline" on page 64). After the node completes migration, you can restart it using the **startrpnode** command (as described in "Bringing the node online in the peer domain" on page 63).

If the peer domain is partitioned, you can fix this problem by stopping all nodes in both sides of the partition and then restarting the peer domain (by using the **startrpdomain** command as described in "Bringing the peer domain online" on page 57) from a node that runs the higher level of RSCT.

Ensuring that network interfaces are included in communication groups when you migrate from versions before 2.4.1.0 or 2.3.5.0

In versions of RSCT before 2.4.1.0/2.3.5.0, when a network interface was marked as down or up by the **ifconfig** command, the configuration resource manager would remove the network interface from, or add it to, the peer domain's communication groups. When you migrate from a version of RSCT before 2.4.1.0 or 2.3.5.0, be aware that RSCT cannot inform if network interfaces that were shut down (and so were excluded from liveness checks that are performed by the communication groups) are later activated. For this reason, after you migrate from a version of RSCT before 2.4.1.0/2.3.5.0, manually set the **HeartbeatActive** persistent attribute of the **IBM.NetworkInterface** resources that were shut down. After you migrate to the new version of RSCT, perform the following actions:

1. On a node that is online in the peer domain, issue the following command to determine if a configuration resource manager considers any of the network interfaces to be down.

   ```
   lsrsrc -a IBM.NetworkInterface
   ```

   The following information is listed for each **IBM.NetworkInterface** resource in the peer domain.

   ```
   resource 4:
           Name             = "eth2"
           DeviceName       = ""
           IPAddress        = "1.1.1.55"
           SubnetMask       = "255.255.255.0"
           Subnet           = "1.1.1.0"
           CommGroup        = "CG2"
           HeartbeatActive  = 0
           Aliases          = {}
           ActivePeerDomain = "ApplDomain"
           NodeNameList     = {"davros.pok.ibm.com"}
   ```

2. If the **IBM.NetworkInterface** attribute of any of the **IBM.NetworkInterface** resources is 0, it indicates that the network interface is excluded from the peer domain's communication groups. To specify that the network interface must be included in the peer domain's communication groups, set the **IBM.NetworkInterface** attribute of the **IBM.NetworkInterface** attribute to 1. For example:

   ```
   chrsrc -a -s 'IPAddress=="1.1.1.55"' IBM.NetworkInterface HeartbeatActive=1
   ```

Migrating an RSCT cluster to the next level

When you migrate an RSCT cluster from one level to the next level, it is recommended that you take the corresponding node or nodes offline in the RSCT peer domain. After the migration is complete, you can bring back the node or nodes online in the RSCT peer domain. If you are migrating to the next level on all, or most of, the nodes in an RSCT peer domain, it is recommended that you take the RSCT peer domain offline, and then bring it back online after the migration is complete.

If you do not take nodes being migrated offline, the nodes might remain pending online or otherwise fail to come online in the peer domain when installation is complete and RSCT subsystems are restarted. If it occurs, you can attempt to recover by forcing the affected nodes offline by using the **forcerpoffline** command as described in “Forcing a peer domain offline” on page 65, bringing the remaining online nodes offline (with the **stoprpdomain** command as described in “Taking individual nodes of a peer domain, or an entire peer domain, offline” on page 64, and then restarting the peer domain (using the **startrpdomain** command as described in “Bringing the peer domain online” on page 57) from a node that is running the higher level of RSCT.

Migrating recently created peer domains from RSCT 2.3.2 to a higher level

A fix that is introduced in RSCT 2.3.3 might cause RSCT peer domains that were created under RSCT 2.3.2 after the start of 2006 to fail to come online when they are migrated to later releases (starting with RSCT 2.3.3). The fix changed the format of the RPD create time that is embedded in the RPD resource identifier. As a result, the resource identifier of a recently created RPD under RSCT 2.3.2 is not recognizable to RSCT 2.3.3 (and later releases) and the domain does not come online. Peer domains that were created under RSCT 2.3.2 in the more distant past (that is, before 2006) must not be affected by this change and must migrate to newer releases without incident.

The following are ways to avoid possible migration problems:

- Do not create new peer domains under RSCT 2.3.2. Instead, migrate to a newer release of RSCT before you create a new domain.
- If you have a recently created peer domain under RSCT 2.3.2 that you must migrate to a newer release of RSCT, contact the IBM Support Center for assistance in devising a unique migration strategy.

Migration considerations for shared secret keys

The shared secret key authentication capability is only available in peer domains with an active version of RSCT that is 2.4.7.1, or later. An existing peer domain can begin only by using shared secret key authentication when all of its nodes are migrated to RSCT 2.4.7.1, or later and the **CompleteMigration** action is run on the domain. Then, a shared secret key can be enabled for the domain as described in "Enabling use of a shared secret key" on page 103.

Also, to ensure the proper operation of shared secret key authentication, all nodes in the peer domain must have a common agreement on the current time of day. The time-of-day clocks on all systems within the peer domain must be synchronized to within a reasonable tolerance of each other, typically a few seconds. If your cluster uses a network time synchronization protocol, such as Network Time Protocol (NTP), the nodes in your peer domain will already have a common agreement on the current time of day; otherwise, use the **date** command on the nodes of your peer domain to make sure their time-of-day clocks are synchronized.

Migrating a peer domain

To complete the migration of a peer domain and update the active version of RSCT to a new level, you must use the **runact** command.

Run this command after all of the nodes defined in the peer domain have been upgraded to a later version. The command only needs to be run once on one of the online nodes with more than half of the nodes online. If all of the upgraded nodes have a version of RSCT that is higher than the currently active version (**RSCTActiveVersion**), the new minimum version of RSCT across all nodes is determined and that version becomes the new active version for the peer domain.

Perform the following steps to complete the migration of a peer domain:

1. Upgrade all of the nodes defined in the peer domain to a later version.
2. Make sure that more than half of the nodes are online. If not, then bring enough nodes online to meet the criteria.
3. Run the following commands on one of the online nodes in the peer domain:
   a. Set the management scope to RSCT peer domain (a value of **2**):
      ```
      export CT_MANAGEMENT_SCOPE=2
      ```
   b. Run the **CompleteMigration** action on the same node to complete the migration of the peer domain. If migrating to a PTF, the PTF must be committed on all nodes before running the **CompleteMigration** action.
      ```
      runact -c IBM.PeerDomain CompleteMigration Options=0
      ```

If you issue the **runact** command before all the nodes have been upgraded or the peer domain has less than half of its nodes online, an error message will result and the **RSCTActiveVersion** will remain unchanged. Upgrade all the nodes to a new level and make sure that half of the peer domain's nodes are online before running the command again.

Migrating a peer domain to a CAA environment

When Cluster-Aware AIX (CAA) is available, an existing peer domain can be migrated so that it operates based on CAA, rather than operating autonomously.

As part of the migration process, a CAA cluster is defined containing the node complement of the peer domain that is being migrated. The interface resource set defined in the **IBM.NetworkInterface** class might change slightly because the view of interfaces on cluster nodes provided by CAA might differ slightly from that acquired by the configuration resource manager from the operating system in a peer domain that is not running in a CAA environment.

Planning the migration of an existing peer domain to a CAA environment

When deciding whether to migrate an existing peer domain to a CAA environment, the constraints that only one CAA cluster can be defined at a time, that there will always be a peer domain corresponding to it defined and online, and that no other peer domains can be defined, are significant. Therefore, consider the following:

- Do not define a CAA cluster before deciding whether any existing peer domain should be migrated to a CAA environment. If a CAA cluster is defined independent of any existing peer domains, a new peer domain corresponding to it is created, and any existing peer domains are deleted.
- If multiple peer domains are defined on a collection of nodes, when CAA is installed and becomes available, only one of those can be migrated to CAA, and the rest will be deleted as part of the migration process. If more than one existing peer domain must be maintained, none can be migrated to CAA.
- When migrating a peer domain to a CAA environment, a physical volume that is shared by all nodes in the peer domain must be provided to support the resulting CAA cluster's repository disk function. Optionally, multiple sets of physical volumes shared by all of the nodes in the peer domain can be identified for application use.

Carrying out migration to a CAA environment

To migrate a peer domain to a CAA cluster after the required shared physical volume sets have been identified (**hdisk1** and **hdisk2**, for example) and the peer domain is online, enter:

```
CT_MANAGEMENT_SCOPE=2 runact -c IBM.PeerDomain MigrateToCAA \
[QuorumType=quorum_type] ReposDisks=hdisk1 \
[SharedDisks=hdisk2,...]
```

If the **QuorumType** attribute is not present, the default value is set to normal or strong quorum (0) in the migrated domain.

This step could take several minutes to complete as the CAA cluster is created and the peer domain nodes run a protocol to switch over to the CAA environment in an orderly fashion. If any part of the migration step fails, the error is reflected in a message and the peer domain continues to operate autonomously as it did prior to migration.

See the *AIX Installation and Migration Guide* for information about migration of an existing CAA cluster to new levels of AIX.

Creating a peer domain

Follow these instructions to configure nodes into an RSCT peer domain.

Before creating a peer domain, review the information in "Prerequisites and restrictions when using configuration resource manager commands" on page 44.

In a peer domain, processor architectures and operating systems are heterogeneous. A peer domain can contain one of these node sets:

- AIX nodes, which support any processor architecture that is supported by the AIX operating system.
- Linux nodes.
- Solaris nodes.
- Windows nodes.
- A combination of AIX nodes and Linux nodes.

Note: AIX nodes and Solaris nodes cannot be part of the same peer domain.

Programs that run in a peer domain might not support the same heterogeneous environment as RSCT. See the specific user documentation for information about supported processor architectures and operating systems.

To configure nodes into an RSCT peer domain, you must first prepare the initial security environment on each node of the peer domain, then create a peer domain definition, and then bring the domain online.

Note: When creating a peer domain that operates on Cluster Aware AIX (CAA) linked clusters, the CAA commands must be used to define the cluster to provide site information.

Related concepts:

"Verifying RSCT installation on Solaris nodes" on page 32
Use the **pkginfo** command to verify that RSCT is installed on a Solaris node.

Related tasks:

"Verifying RSCT installation on the Windows platform" on page 34
RSCT is automatically installed when you install Tivoli System Automation on a Windows system.

Related reference:

"Supported Linux distributions for RSCT" on page 31
A number of Linux operating system distributions are supported by RSCT.

Preparing the initial security environment on each node

Prepare the initial security environment on each node before creating a peer domain using the **mkrpdomain** command.

Before you can create your peer domain using the **mkrpdomain** command (described in "Creating a peer domain definition" on page 52), you first need to run the **preprpnode** command to establish the initial trust between each node that will participate in the peer domain, and the node from which you will run the **mkrpdomain** command. Later, when you run the **mkrpdomain** command, the configuration resource manager will establish the additional needed security across all peer domain nodes. This will enable you to issue subsequent commands from any node in the peer domain.

Note: The **preprpnode** command will automatically exchange public keys between nodes. If you do not feel the security of your network is sufficient to prevent address and identity spoofing, see "Guarding against address and identify spoofing when transferring public keys" on page 276. If you are not sure if your network is secure enough, consult with a network security specialist to see if you are at risk.

This preparatory step is not needed when you create a peer domain in a CAA environment because security arrangements are handled separately from peer domain functions.

The node from which you will issue the **mkrpdomain** command is called the *originator node*. Be aware that the originator node does not have to be a node you intend to include in your RSCT peer domain; it could be just a node from which you issue the **mkrpdomain** command. It could, for example, be the management server of a management domain. To establish trust between the originator node and each node that will be in the peer domain, you must run the **preprpnode** command on each node that will be in the peer domain. You will need to specify the name of the originator node as the parameter.

For example, suppose that you will be issuing the **mkrpdomain** command on *nodeA*. From each node that will be in the peer domain, issue the command:

```
preprpnode nodeA
```

You can also specify multiple node names on the command line:

```
preprpnode nodeA nodeB
```

Instead of listing the node names on the command line, you can, using the **-f** flag, specify the name of a file that lists the node names. For example:

```
preprpnode -f node.list
```

When using the **preprpnode** command, you can identify the node by its IP address or by the long or short version of its Domain Name System (DNS) name. If any IP address for the originator node cannot

be resolved to a DNS name, than all IP addresses associated with the originator node should be specified on the **preprpnode** command. This enables you to specify an IP address that is not DNS resolvable on the **mkrpdomain** command (as described in "Creating a peer domain definition"). If you are certain that all IP addresses you will later specify on the **mkrpdomain** command will be resolvable to DNS names, then it is not necessary to specify all of the originator node's IP addresses on the **preprpnode** command. In this case, however, if you do identify the originator node by an IP address, you must be certain that the IP address is resolvable to a DNS name.

The **preprpnode** command establishes the initial security environment needed by the **mkrpdomain** command by:

- ...retrieving the originator node's public key and adding it to the trusted host list of the local node.
- ...modifying the local node's RMC access control list (ACL) to enable access to its resources from the originator node.

You can specify multiple nodes on the **preprpnode** command, in which case the initial trust will be established between the local node and each of the remote nodes listed. As long as you know which node will be the originator node, however, there should not be a need to specify multiple nodes on the **preprpnode** command.

If you have, for security reasons, already manually transferred the public keys, you need to use the **-k** flag when you issue the **preprpnode** command. For example:

```
preprpnode -k nodeA nodeB
```

Using the **-k** flag disables the automatic transfer of public keys. While allowing the **preprpnode** command to copy the public key again will not result in an error, you could reduce overhead by disabling the transfer.

Although the **-k** flag disables automatic public key transfer, the **preprpnode** command will still modify the node's RMC ACL file to enable access to the other nodes you will include in the peer domain.

For complete syntax information on the **preprpnode** command, see the *Technical Reference: RSCT for AIX* or the *Technical Reference: RSCT for Multiplatforms*.

Once you have run the **preprpnode** command on each node that you will include in the peer domain, you can create a new peer domain.

Related information:

preprpnode Command

mkrpdomain Command

Creating a peer domain definition

Use the **mkrpdomain** command to create a peer domain definition.

A peer domain definition must consist the following items:

- A peer domain name
- A list of nodes that are included in the peer domain

Optionally, a peer domain definition can also consist of the following items:

- The quorum type to be used to calculate startup quorum, configuration quorum, and operational quorum.
- The User Datagram Protocol (UDP) port numbers to be used for Topology Services and Group Services daemon-to-daemon communication.
- The fanout value, or maximum number of threads the configuration resource manager can use to perform parallel operations.
- The shared secret key type, if any, to be used to authenticate control messages for the domain.

- Node list information that specifies which nodes must or must not be quorum nodes or preferred nodes. If this information is not specified, by default, all nodes are quorum nodes and preferred nodes.
- Policy information that determines whether the **Name** and **HostName** attributes of the **IBM.PeerNode** class must be synchronized.

You can use these ASCII characters in your domain name: A to Z, a to z, 0 to 9, - (hyphen), . (period), and _ (underscore). The first character of the domain name cannot be a hyphen.

When you issue the **mkrpdomain** command, the configuration resource manager also creates the communication group definitions that are needed later to enable liveness checks (known as heartbeats in Topology Services) between the nodes of a peer domain. The configuration resource manager automatically attempts to form a communication group that is based on subnets and inter-subnet accessibility. Each communication group is identified by a unique name. The name is assigned sequentially by suffixing `CG` with existing highest suffix + 1, such as `CG1`, `CG2`, and so on.

When you run the **startrpdomain** command, the configuration resource manager supplies the communication group definition information to Topology Services.

Quorum type

If the quorum type is not specified in the **mkrpdomain** command, a default quorum type is used to calculate startup quorum, configuration quorum, and operational quorum. The default quorum type depends on your environment. For most clusters, the default quorum type is `Normal`. In IBM i environments, the default is `Override`. In environments with the IBM TotalStorage SAN File System, the default is `SANFS`. To specify a quorum type, you can use the **-Q** flag followed by an integer or name that indicates the quorum type. In the **mkrpdomain** command, you can specify the quorum type to be one of the following types:

- 0 or `Normal`
- 1 or `Quick`

Note: The quorum types 3 (`Override`) and 4 (`SANFS`) are defined only for a few dedicated and embedded environments. You do not need to explicitly set the quorum type to either of these values.

When you start a peer domain, you can override the quorum type to specify a different quorum type for calculating startup quorum.

UDP port numbers

If the port numbers are not specified for Topology Services and Group Services daemon to daemon communication in the **mkrpdomain** command, the default port numbers (port 12347 for Topology Services and port 12348 for Group Services) are used. You can override these defaults by using the **-t** flag (to specify the Topology Services port) or **-g** flag (to specify the Group Services port). Any unused port in the range 1024 - 65535 can be assigned.

Fanout value

For many of its operations on a peer domain, the configuration resource manager must establish a number of connections to remote nodes. In the case of the **mkrpdomain** command, for example, a remote connection is made to each node in the domain that is being formed. Subsequent operations, such as bringing the peer domain online, must make similar connections. The configuration resource manager uses a number of threads to carry out such remote operations in parallel. By default, the maximum number of threads that the configuration resource manager uses is 128. If more remote connections are needed than available threads, the configuration resource manager waits for threads to be available to connect with other remote nodes. For large clusters, performance can be improved by increasing the

maximum number of threads that the configuration resource manager uses to perform parallel operations. The maximum number of threads is called the fanout value and can be specified by using the **-m** flag of the **mkrpdomain** command.

The fanout value that is specified on the **mkrpdomain** command is applied to all subsequent operations in the peer domain, unless explicitly overridden or reset. The fanout value is overridden and reset in the following ways:

- You can override the fanout value when you start a peer domain by using the **startrpdomain** command. However, the fanout value that is specified on the **startrpdomain** command applies to that startup operation only.
- You can reset the fanout value by using the **chrsrc** command to modify the **Fanout** attribute of the **IBM.PeerNode** resource class.

Shared secret key

By using RSCT shared secret keys, RSCT components can determine the authenticity of control messages that they receive from their peers and detect any alteration of these control messages. Shared secret keys are distinct from the public or private key pairs that are exchanged when you prepare to create a peer domain. Control messages are signed by the sender by using the shared secret key, allowing the recipient to verify the authenticity of the sender and the integrity of the messages. An active shared secret key for a peer domain exists only in the context of that domain, when the domain is online. The Topology Services, Group Services, and resource monitoring and control (RMC) components of RSCT use the shared secret key. The configuration resource manager manages the active key automatically.

Message signatures are created for control messages by using the MD5 hash algorithm and then encrypting the hash by using the shared secret key for protection. The following table lists the types of shared secret keys that are supported for peer domains in order of their increasing strength.

Table 17. RSCT shared secret key types

Key type	Description
CSSKTYPE_DES_MD5	DES encryption that uses a 56-bit key
CSSKTYPE_3DES_MD5	Triple DES encryption that uses a 168-bit key
CSSKTYPE_AES256_MD5	AES encryption that uses a 256-bit key

By default, no shared secret key is active for a new or migrated peer domain. You must explicitly enable this feature by using the **-k** flag of the **mkrpdomain** command with one of the key types.

When the domain is created, the configuration resource manager automatically prepares it to use the specified type of shared secret key for message authentication and use of the key is automatically initiated when the domain is started with the **startrpdomain** command.

The configuration resource manager automatically refreshes an active shared secret key in a peer domain on a regular interval whenever the peer domain is online. The default refresh interval is one day. To specify a different refresh interval (or no automatic refresh at all), you can specify the **-r** flag in the **mkrpdomain** command along with the required refresh interval expressed as `dd:hh:mm:ss` (in days, hours, minutes, and seconds), or 0 if no automatic refresh is required. Because the secret key is shared among all hosts in the peer domain, it is automatically changed on a regular interval to make key guessing more difficult and less viable, depending on the length of the refresh interval. For instance, this interval can be based on the message traffic analysis of the peer domain. Refreshing or changing the shared secret key is analogous to a password change policy, except that the key is automatically generated, distributed, and used within the context of the active peer domain.

Note: To prevent control messages from timing out and to ensure the proper operation of shared secret key authentication, all nodes in the peer domain must have a common agreement on the current time of

day. The time-of-day clocks on all systems within the peer domain must be synchronized to within a reasonable tolerance of each other, typically, only a few seconds. If your cluster utilizes a network time synchronization protocol, such as NTP, then the nodes in your peer domain already have a common agreement on the current time of day; otherwise, use the date command on the nodes of your peer domain to ensure that their time-of-day clocks are synchronized.

Failure to synchronize the time-of-day clocks across all nodes in the peer domain can cause the domain to remain in the pending online state after you issue the **startrpdomain** command.

Policy

In a CAA environment, by default, RSCT uses the **HostName** attribute of the **IBM.PeerNode** class for the **Name** attribute. However, in many cases, the **Name** attribute must not change dynamically. This synchronization between the **Name** and **HostName** attributes is controlled by a policy.

You can specify the policy information by using the **-p** flag in the **mkrpdomain** command. Valid values for policy are 0 and 1. If the policy value is 1, the **Name** attribute of the **IBM.PeerNode** class is maintained in sync with the **HostName** attribute of the **IBM.PeerNode** class. If the policy value is 0, the **Name** attribute of the **IBM.PeerNode** class is not maintained in sync with the **HostName** attribute.

For CAA environment, if the **-p** flag is not specified in the **mkrpdomain** command, the default value of 1 is set for the policy attribute. For non-CAA environment, the default value of 0 is set for the policy attribute.

After you create your peer domain definition, you can bring the peer domain online.

Related information:

mkrpdomain Command

Examples for creating a peer domain definition:

The **mkrpdomain** command can be used with different flags depending on your requirements to create a peer domain definition.

The following list of examples describe the various ways of creating a peer domain definition in different scenarios:

1. To create a peer domain with multiple nodes:

   Scenario: You want to establish a peer domain with three nodes and the nodes are identified by the DNS names *nodeA*, *nodeB*, and *nodeC*. When you issued the **preprpnode** command from the nodes that make up your peer domain, you determined that *nodeA* is the originator node. In this case, to create a peer domain named `ApplDomain`, run the following command from *nodeA*:

   ```
   mkrpdomain ApplDomain nodeA nodeB nodeC
   ```

   This command creates the peer domain definition `ApplDomain` that consists the nodes *nodeA*, *nodeB*, and *nodeC*.

2. To create a peer domain by specifying a file that contains the list of all the node names:

   Scenario: You want to create a peer domain with many nodes. Instead of manually listing the node names on the command line, you want to specify the name of a file that contains the list of node names. To do so, run the following command:

   ```
   mkrpdomain -f node.list ApplDomain
   ```

   This command creates the peer domain definition `ApplDomain` that consists of the nodes that are listed in the `node.list` file.

3. To use the IPv6 addresses as resources for heartbeating and internal peer domain operations:

Scenario: The nodes in your peer domain (`ApplDomain`) use IPv4 addresses. IPv6 addresses are also configured on the network interfaces of the nodes. To see these IPv6 addresses being represented as resources in the **IBM.NetworkInterface** class and to use them for heartbeating and internal peer domain operations, run the following command:

```
mkrpdomain -6 ApplDomain
```

This command establishes a peer domain in which the IPv6 addresses that are configured on the nodes' network interfaces are visible as resources in the **IBM.NetworkInterface** class. They are formed into their own communication groups for use in heartbeating and internal peer domain operations. If the **-6** flag is not specified, no IPv6 addresses are visible as resource in the **IBM.NetworkInterface** class or used in this way.

4. To prevent the **mkrpdomain** command to fail for all the nodes that are based on a single node failure:

   Scenario: You are creating a large peer domain configuration, in which the `node.list` file contains many nodes. In such a case, the chances of the **mkrpdomain** command failure on any one node is greater. By default, if the **mkrpdomain** command fails on any node, it fails for all nodes. You want to override this default behavior such that the **mkrpdomain** command does not fail for all nodes that are based on a single node failure. You can do so by running the following command:

   ```
   mkrpdomain -c -f node.list ApplDomain
   ```

5. To create a peer domain definition with `Quick` quorum type:

   Scenario: You are creating a large peer domain configuration. You want to specify the quick startup mode that is helpful in creating such large clusters. You can do so by running the following command:

   ```
   mkrpdomain -Q 1 ApplDomain nodeA nodeB nodeC
   ```

   Or,

   ```
   mkrpdomain -Q Quick ApplDomain nodeA nodeB nodeC
   ```

6. To specify port 1200 for Topology Services and port 2400 for Group Services:

   Scenario: You want to override the default ports with the value of 1200 for the Topology Services and 2400 for the Group Services. You can do so by running the following command:

   ```
   mkrpdomain -t 1200 -g 2400 ApplDomain nodeA nodeB nodeC
   ```

7. To specify a fanout value:

   Scenario: You are creating a large peer domain configuration. You want to specify a larger fanout value of 900 to improve performance. You can do so by running the following command:

   ```
   mkrpdomain -m 900 -f node.list ApplDomain
   ```

8. To reset the fanout value:

   Scenario: You want to reset the fanout value to 1000 so that the earlier fanout value is not applied to all the subsequent operations in the peer domain. You can do so by running the following commands from a node that is online in the peer domain:

   a. Set the management scope to the RSCT peer domain scope:

      ```
      export CT_MANAGEMENT_SCOPE=2
      ```

   b. Issue the following **chrsrc** command:

      ```
      chrsrc -c IBM.PeerNode Fanout=1000
      ```

9. To enable a shared secret key type of `CSSKTYPE_DES_MD5`:

   Scenario: You want to use the `CSSKTYPE_DES_MD5` shared secret key in your peer domain so that the RSCT component can verify the authenticity of the control messages that they receive. You can do so by running the following command:

   ```
   mkrpdomain -k CSSKTYPE_DES_MD5 ApplDomain nodeA nodeB nodeC
   ```

10. To specify a shared secret key refresh interval:

Scenario: You want the shared secret key to be refreshed more frequently than the default 24 hours. So, you want to specify a refresh interval of 12 hours. You can do so by running the following command:

```
mkrpdomain -k CSSKTYPE_DES_MD5 -r 12:00:00 ApplDomain nodeA nodeB nodeC
```

11. To specify the policy information:

    Scenario: You are creating a peer domain `ApplDomain` in a CAA environment and you want to specify the policy information such that the **Name** (`NodeA`) and **HostName** (`host_nameA`) attributes must not be in sync with each other. You can do so by running the following command:

    ```
    mkrpdomain -p 0 ApplDomain nodeA@host_nameA
    ```

Bringing the peer domain online

The **startrpdomain** command brings a peer domain online by starting the resources on each node belonging to the peer domain.

To bring the peer domain online, pass the **startrpdomain** command the name of a peer domain you have already defined using the **mkrpdomain** command. For example, to bring the peer domain *ApplDomain* online, you would, from any of the nodes in the peer domain, issue the command:

```
startrpdomain ApplDomain
```

The peer domain's quorum type will determine the startup quorum needed for bringing the peer domain online. The cluster's quorum type will either be the default for your environment, or one you specified using the **mkrpdomain** command's **-Q** flag. When starting a peer domain, you can also, if the quorum type is set to 0 (Normal) or 1 (Quick), override the quorum type to specify a different one for calculating startup quorum. Using the **startrpdomain** command's **-Q** flag, you can specify the startup quorum type to be either:

- 0 or Normal
- 1 or Quick

For example, if the quorum type is 0 (Normal), you could override that quorum type to specify the use of quick startup mode to calculate startup quorum.

```
startrpdomain -Q 1 ApplDomain
```

or

```
startrpdomain -Q Quick ApplDomain
```

Notes:

1. You cannot modify the startup quorum type if it has been implicitly set to 2 (Override) or 3 (SANFS).
2. You cannot specify the startup quorum type to be 2 (Override) or 3 (SANFS).

When bringing the peer domain online, the **startrpdomain** command uses the peer domain configuration information you defined when you issued the **mkrpdomain** command. If necessary, the configuration resource manager will start Group Services and Topology Services on each of the nodes in the peer domain. The configuration resource manager will also supply Topology Services with the communication group definition information for the peer domain. A communication group controls how liveness checks (*heartbeating* in Topology Services) are performed between the communications resources within the peer domains. The communication group also determines which devices are used for heartbeating in the peer domain. Each communication group has several characteristics. These characteristics specify:

- the number of missed heartbeats that constitute a failure
- the number of seconds between the heartbeats
- whether or not broadcast should be used
- whether or not source routing should be used

Each communication group also has a list of its member network interfaces.

To determine what communication groups were created, use the **lscomg** command (as described in "Listing communication groups" on page 71). The **lscomg** command not only lists the communication groups in your peer domain but also shows the characteristics about those communication groups. This means that even if the communication group was created automatically, you can use the **lscomg** command to see its default characteristics. If you would like to modify any of these characteristics, you can use the **chcomg** command as described in "Modifying a communication group's characteristics" on page 73. To modify network interfaces in the communication group, see "Modifying a communication group's network interface" on page 75.

By default, the **startrpdomain** command will not attempt to bring the peer domain online until at least half the nodes have been contacted. The configuration resource manager searches for the most recent version of the peer domain configuration which it will use to bring the peer domain online. If you want the configuration resource manager to contact all nodes in the peer domain before bringing the domain online, specify the **startrpdomain** command's **-A** flag. This option is useful if you want to be sure that the most recent configuration is used to start the peer domain. For example:

```
startrpdomain -A ApplDomain
```

If you want the configuration resource manager to get the most recent configuration information from the local node only, specify the **startrpdomain** command's **-L** flag. For example:

```
startrpdomain -L ApplDomain
```

The configuration resource manager will not try to contact nodes to determine the latest configuration beyond a specified timeout value which is, by default, 120 seconds. If at least half the nodes (or all nodes if you have specified the **-A** flag) have not been contacted in that time, the configuration resource manager will not start the peer domain. You can, however, increase the timeout value using the **startrpdomain** command's **-t** flag. For example, to have the operation time out at 240 seconds, you would issue the command:

```
startrpdomain -t 240 ApplDomain
```

When bringing a peer domain online, the configuration resource manager needs to establish a number of connections to remote nodes, and will use a number of threads to carry out these operations in parallel. The maximum number of threads the configuration manager will use is called the *fanout value*, and was either specified by the **mkrpdomain** command's **-m** flag, specified using the **chrsrc** command, or is the default value of 128. To ascertain the current fanout value for the peer domain, you can issue the following command:

```
lsrsrc -c IBM.PeerNode Fanout
```

The following output is displayed:

```
Resource Class Persistent Attributes for IBM.PeerNode
resource 1:
        Fanout = 128
```

When issuing the **startrpdomain** command, you can use the **-m** flag to specify the fanout value for this startup operation. For example:

```
startrpdomain -m 900 ApplDomain
```

After the domain is brought online, you can use the **lsrpnode** command to list information about the nodes in the domain. You can run this command from any node in the peer domain. The following results are displayed:

```
Name                    OpState                 RSCTVersion
nodeA                   online                  3.1.5.0
nodeB                   online                  3.1.5.0
nodeC                   online                  3.1.5.0
nodeD                   offline                 3.1.5.0
nodeE                   offline                 3.1.5.0
```

You can also view all the network interfaces in the domain by issuing the **lsrsrc** command. Before issuing this generic RMC command, you should first set the management scope to 2 to indicate it is a peer domain, as follows:

```
export CT_MANAGEMENT_SCOPE=2
```

Then you can view the network interfaces in the peer domain by issuing the command:

```
lsrsrc -a IBM.NetworkInterface
```

Note: When you use the **-a** flag on the **lsrsrc** command, the **lsrsrc** command will automatically set the CT_MANAGEMENT_SCOPE environment variable. Explicitly set the CT_MANAGEMENT_SCOPE environment variable only if the node is in both a peer domain and a management domain.

When a node becomes a member of the peer domain, it is assigned a unique integer which is referred to as a *node number*. Node numbers are used on certain commands and by some subsystems (for example, Topology Services). To view the node numbers, issue the following command from any online node in the peer domain. The attribute NodeList identifies the node numbers of all the nodes defined in the online cluster.

```
lsrsrc -a IBM.PeerNode Name NodeList
```

Note: We recommend that, once a peer domain is created and the peer nodes are online, you save a record of the node to node number mapping. Such a record may be helpful if you later need to restore nodes with their original node numbers (as described in *Troubleshooting RSCT*). To save a record of the node to node number mapping, issue the following command from a node that is online in the peer domain.

```
lsrsrc -x -D' ' IBM.PeerNode Name NodeList | sed 's/{/ /g' | \
sed 's/}/ /g'|sed 's/"//g' > rpdNodeMap.save
```

You can later take the peer domain offline using the **stoprpdomain** command. You can also take an individual node offline using the **stoprpnode** command.

For complete syntax information on the **startrpdomain** command, see its man page in the *Technical Reference: RSCT for AIX* or the *Technical Reference: RSCT for Multiplatforms*.

Related information:

startrpdomain Command

Creating a peer domain in a CAA environment

Cluster-Aware AIX (CAA) and RSCT maintain an equivalence between a CAA cluster and the peer domain representing it.

If a CAA cluster is created by using CAA interfaces or commands, the corresponding peer domain is created automatically. You can also use the **mkrpdomain** command to create a CAA cluster and the corresponding peer domain. A CAA cluster depends on a physical volume shared across all of the nodes in the cluster to support the central repository in which it stores its configuration, and might optionally support a second set of shared physical volumes for application use. After these volumes have been made available (*pv_1* and *pv_set_2* in this example), the peer domain and cluster can be created by using this command:

```
mkrpdomain -C 1 -R pv_1 [ -D pv_set_2 ][-Q quorum_type | quorum_type_name] domain_name node1 ...
```

The default and the only supported value of **QuorumType** to the **mkrpdomain** command for CAA is `QuorumLess` (4). After the domain is online, the **QuorumType** attribute can be changed by using the following command:

```
> chrsrc –c IBM.PeerNode QuorumType=new_quorum_type
```

Whether the CAA cluster and peer domain are created by using CAA commands or the **mkrpdomain** command, both methods result in the peer domain and cluster definitions being established and

corresponding to each other. In addition, the peer domain is brought to the online state on all nodes in the configuration that are currently operating. It is different than a peer domain not operating on CAA, where the peer domain is brought online separately from when it is created. Furthermore, the peer domain corresponding to a CAA cluster remains defined and online for the life of the cluster. There is no need for the preprpnode step to share public keys among nodes before creating a peer domain on CAA.

Adding nodes to an existing peer domain

After you create a peer domain, you can add new nodes to it.

"Creating a peer domain" on page 50 describes the initial setup of a peer domain. To add a node to the existing peer domain, you must first prepare the security environment on the node. After you add the node, you can bring the node online.

Preparing the security environment on the node

Before you can add a node to a peer domain using the **addrpnode** command, you must first issue the **preprpnode** command to establish the initial trust between the node to be added, and the node from which you will issue the **addrpnode** command. Later, when you issue the **addrpnode** command, the configuration resource manager will establish the additional security environment so that the new node can issue subsequent configuration resource manager commands. This preparatory step is not needed when you add a node to a peer domain that is operating in a CAA environment because security arrangements are handled separately from peer domain functions.

The node from which you will issue the **addrpnode** command is called the *originator node*, and must be a node that is already part of the RSCT peer domain. To establish trust between the originator node and the node to be added to the peer domain, you must first run the **preprpnode** command on the node to be added. On the **preprpnode** command, you must either specify all the existing nodes in the peer domain, or else you must specify the configuration resource manager group leader. If the peer domain does not consist of many nodes, you will probably find it easiest to specify all the existing nodes on the **preprpnode** command. For example, if the peer domain consists of *nodeA*, *nodeB*, and *nodeC*, you would enter the following on the node you wish to add to the peer domain:

```
preprpnode nodeA nodeB nodeC
```

You identify the nodes by their IP addresses or by the long or short version of their DNS names. If you will later specify an IP address on the **addrpnode** command that cannot be resolved to a DNS name, then you must specify all IP addresses of all nodes (or all IP addresses of the configuration resource manager group leader only) on the **preprpnode** command. If you are certain that the IP address you will later specify on the **addrpnode** command will be resolvable to a DNS name, then you do not need to specify all IP addresses on the **preprpnode** command. In this case, however, if you do identify any node on the **preprpnode** command by its IP address, you must be certain that the IP address is resolvable to a DNS name.

If you are unsure which nodes are in a peer domain, enter the **lsrpnode** command from a node that is active in the peer domain.

```
lsrpnode
```

Output is similar to:

```
Name    OpState  RSCTVersion
nodeA   Online   3.1.5.0
nodeB   Online   3.1.5.0
nodeC   Online   3.1.5.0
```

Instead of listing the node names on the command line, you can, using the **-f** flag, specify the name of a file that lists the node names or IP addresses. When the peer domain consist of a large number of nodes, you may find listing the nodes in a file easier than entering them all on the command line. For example, if the nodes were listed in the file *node.list*, you would enter the following command on the node you will be adding to the peer domain:

```
preprpnode -f node.list
```

An easy way to generate the *node.list* file used in the preceding example, would be to enter the following command on a node that is online in the peer domain:

```
lsrpnode -x | awk '{print $1}' > node.list
```

Once the file is generated, send it to the new node on which you will enter the **preprpnode** command.

Another method that you may find easier when adding a node to a large peer domain, is to specify the peer domain's group leader on the **preprpnode** command. Specifying the group leader eliminates the need to specify all the nodes in the peer domain. *Group leader* is a Topology Services and Group Services term for a coordinating node of a configuration resource manager group. Peer domains use the Topology Services and Group Services subsystems for distributed coordination and synchronization.

To find out which node in the peer domain is the group leader, enter the following SRC command on a node that is online in the peer domain:

```
lssrc -ls IBM.ConfigRM
```

The following results are displayed. Make note of the group leader (highlighted in bold text in this example).

```
Subsystem         : IBM.ConfigRM
PID               : 17880
Cluster Name      : Zagreus
Node Number       : 1
Daemon start time : Mon Oct 20 22:01:43 EDT 2011

Daemon State: Online in JoeD

ConfigVersion: 0x53fb2ff09
Group IBM.ConfigRM:
   Providers: 2
   GroupLeader: node8, 0x9a6befe2be807d07, 1

Information from malloc about memory use:
   Total Space    : 0x009c0480 (10224768)
   Allocated Space: 0x0086fad8 (8846040)
   Unused Space   : 0x0014e3e0 (1369056)
   Freeable Space : 0x00000000 (0)
```

Supply the name of the group leader node on the **preprpnode** command:

```
preprpnode node8
```

Specifying the group leader node eliminates the need to specify the other nodes in the peer domain.

The configuration resource manager and Group Services apply slightly different concepts when determining which node is the group leader. The configuration resource manager group leader is the node on which the configuration resource manager daemon runs actions that trigger a Group Services protocol, to modify the peer domain configuration, for example. The Group Services group leader is the node on which the Group Services daemon coordinates execution of such protocols. Occasionally, a single node might be the configuration resource manager group leader and the Group Services group leader, particularly when the peer domain is first started. Over time, however, the node serving as the group leader may differ across the domain's nodes because of various events specific to the configuration resource manager or Group Services daemon populations, such as daemon restart.

Knowledge of the Group Services name server and group leader is useful when you want to constrain them to "preferred" nodes. See the *Troubleshooting RSCT* for details on how to determine which node is the Group Services name server and group leader and “Defining preferred nodes for Group Services name server and group leader selection” on page 83 for information on Group Services preferred nodes.

If you have chosen, for security reasons, to manually transfer the public keys, use the **-k** flag when you issue the **preprpnode** command. For example:

```
preprpnode -k nodeA nodeB nodeC
```

Using the **-k** flag disables the automatic transfer of public keys. While allowing the **preprpnode** command to copy the public key again will not result in an error, you could reduce overhead by disabling the transfer.

Although the **-k** flag disables the public key transfer, the **preprpnode** command will still modify the node's RMC ACL file to enable access to the other nodes in the peer domain.

For more information about the **preprpnode** command, see the *Technical Reference: RSCT for AIX* or the *Technical Reference: RSCT for Multiplatforms*.

Once you have set up the security environment on the node, you can add it to the peer domain.

Adding nodes to the peer domain

Use the **addrpnode** command to add one or more nodes to an RSCT peer domain definition, passing it the IP address or DNS name of the node you want to add.

When you initially set up an RSCT peer domain, you use the **mkrpdomain** command to create the initial peer domain definition. To add one or more nodes to that existing peer domain definition, use the **addrpnode** command, passing it the IP address or DNS name of the node you want to add. However, any change to the online cluster definition requires a configuration quorum of $(n/2) + 1$ nodes (where n is the number of nodes that are defined in the cluster) to be active. In other words, you cannot change an online cluster definition unless a majority of the nodes are online in the domain.

The configuration resource manager then modifies the communication group definitions that are needed later to extend liveness checks to the new nodes. This liveness check is called as heartbeating in Topology Services. When you issue the **startrpnode** command, the configuration resource manager supplies the modified communication group definition information to Topology Services.

Examples

1. To add the node that has the DNS name of `nodeD` to a peer domain, run the following command from a node in the peer domain:

   ```
   addrpnode nodeD
   ```

2. To add multiple nodes to the peer domain definition, run the following command:

   ```
   addrpnode nodeD nodeE
   ```

3. To add many nodes to the peer domain definition, when it becomes difficult to specify all the nodes in the command line manually, run the following command:

   ```
   addrpnode -f node.list
   ```

   where, the `node.list` file contains the list of the node names to be added. Optionally, the `node.list` file can also contain the information as to which nodes must be the quorum or preferred nodes and vice versa.

4. To add multiple nodes to the peer domain definition along with the host names, run the following command:

   ```
   addrpnode nodeA@host_nameA nodeB@host_nameB
   ```

   where, `host_nameA` is the host name of the node `nodeA` and `host_nameB` is the host name of the node `nodeB`.

After you add a node to an existing peer domain definition, you can bring the node online.

Related information:

"Working with communication groups" on page 70
Several optional tasks are available when working with communication groups.

"The Topology Services subsystem" on page 286
In an RSCT peer domain, the configuration resource manager uses the Topology Services subsystem to monitor the liveness of the adapters and networks included in communication groups.

addrpnode Command

Bringing the node online in the peer domain

The **startrpnode** command brings an offline node online in the current peer domain.

To determine which nodes are currently defined in the peer domain, use the **lsrpnode** command from any node in the peer domain.

```
lsrpnode
```

Issuing this command lists information about the nodes defined in the peer domain. For example:

```
Name                OpState             RSCTVersion
nodeA               online              3.1.5.0
nodeB               online              3.1.5.0
nodeC               online              3.1.5.0
nodeD               offline             3.1.5.0
nodeE               offline             3.1.5.0
```

In this example, *nodeD* and *nodeE* are currently offline. Before you bring them online in the current RSCT peer domain, you might want to check that the nodes are not online in another RSCT peer domain. A node can be defined to more than one peer domain, but can be online in only one at a time. If you issue the **startrpnode** command for a node that is already online in another peer domain, the node is not brought online in the new peer domain, but instead remains online in the other peer domain. To list peer domain information for a node, use the **lsrpdomain** command. For example, to determine if *nodeD* is currently online in any other peer domain, issue the following command on *nodeD*:

```
lsrpdomain
```

Issuing this command lists information about the peer domains a node is defined in. For example:

```
Name          OpState     RSCTActiveVersion    MixedVersions    TSPort   GSPort
ApplDomain    offline     3.1.5.0                no               12347    12348
```

This output shows us that *nodeD* is not defined in any other peer domain, and so cannot be online in any other peer domain. To bring it online in the current peer domain, issue the command from any online node:

```
startrpnode nodeD
```

Note: The **startrpnode** command does not work in the CAA environment.

The configuration resource manager, at this time, supplies Topology Services on the new node with the latest cluster definition for the peer domain. It extends the Topology Services liveness checks to the new node.

If there are multiple nodes offline in the peer domain, you can also use the **startrpdomain** command to bring all of the offline nodes online in this peer domain. For example, to bring the peer domain *ApplDomain* online, you would, from any node, issue the command:

```
startrpdomain ApplDomain
```

All the offline nodes, if not already online in another peer domain, are invited to go online.

For more information about the **startrpdomain** command, see the directions for creating a peer domain (the **startrpdomain** command is described in more detail in "Bringing the peer domain online" on page 57

57 of those directions). For complete syntax information about the **startrpnode**, **startrpdomain**, **lsrpnode**, or **lsrpdomain** commands, see their man pages in the *Technical Reference: RSCT for AIX* or the *Technical Reference: RSCT for Multiplatforms*.

Adding nodes to a peer domain in a CAA environment

Adding nodes to a peer domain in a CAA environment is almost equivalent to adding nodes in a peer domain that is not running in a CAA environment. In a CAA environment, you can add nodes to a peer domain by using the **addrpnode** command and the node becomes online in the peer domain shortly after it is added. The **preprpnode** step is not needed in a CAA environment.

The node is added to the peer domain and the CAA cluster. The CAA cluster that has up to 32 nodes is supported by RSCT version 3.1.5.0, and later.

Taking individual nodes of a peer domain, or an entire peer domain, offline

You might want to take an entire peer domain or its individual nodes offline in order to perform node maintenance or make application upgrades.

To perform either of these tasks use the following commands:

- Use the **stoprpnode** command to take a peer domain node offline
- Use the **stoprpdomain** command to take a peer domain offline

Taking a peer domain node offline

The **stoprpnode** command takes one or more nodes of a peer domain offline.

You might need to do this task to perform application upgrades, to perform maintenance on a node, or before removing the node from the peer domain. Also, since a node might be defined in multiple peer domains, but online in only one at a time, you might need to take a node offline in one peer domain so that you might bring it online in another. To take a node offline, issue the **stoprpnode** command from any online node in the peer domain, and pass it the peer domain node name of the node to take offline.

You can list the peer domain node names by issuing the **lsrpnode** command for any node in the peer domain:

```
lsrpnode
```

Issuing this command lists information about the nodes defined in the peer domain. This information includes the peer domain node names. For example:

```
Name                OpState             RSCTVersion
nodeA               offline             3.1.5.0
nodeB               online              3.1.5.0
nodeC               online              3.1.5.0
nodeD               online              3.1.5.0
nodeE               offline             3.1.5.0
```

To take the node whose peer domain node name is *nodeA* offline, you would issue the following command from any online node:

```
stoprpnode nodeA
```

Note: The **stoprpnode** command does not work in the CAA environment.

You can also take multiple nodes offline. For example:

```
stoprpnode nodeA nodeB
```

An RSCT subsystem (such as Topology Services or Group Services) might reject the **stoprpnode** command request to take a node offline if a node resource is busy. To force the RSCT subsystems to take the node offline regardless of the state of node resources, use the **stoprpnode** command **-f** flag. For example:

```
stoprpnode -f nodeA
```

To later bring the node back online, use the **startrpnode** command.

Taking a peer domain offline

You might wish to take a peer domain offline to perform maintenance.

To take a peer domain offline, issue the **stoprpdomain** command from any online node in the peer domain. You pass the **stoprpdomain** command the name of the peer domain you want to take offline. For example, to take all the nodes in the peer domain *ApplDomain* offline:

```
stoprpdomain ApplDomain
```

An RSCT subsystem (such as Topology Services or Group Services) might reject the **stoprpnode** command's request to take a peer domain offline if a peer domain resource is busy. To force the RSCT subsystems to take the peer domain offline regardless of the state of peer domain resources, use the **stoprpdomain** command's **-f** flag. For example:

```
stoprpdomain -f ApplDomain
```

Stopping a peer domain does not remove the peer domain definition; the peer domain can therefore be brought back online using the **startrpdomain** command.

Forcing a peer domain offline

When trying to bring a node online in a peer domain while the domain is operating under quorum, it is possible for the node to remain in the pending online state indefinitely. To remedy the condition, you can issue the **forcerpoffline** command on the node that is in the pending online state.

For example:

```
forcerpoffline domain_name
```

The **forcerpoffline** command causes the configuration resource manager and the RMC subsystem to be recycled. The command must be run by root on the node that is in the pending online state. If you do not know why the node is remaining in the pending online state, use **ctsnap** to capture a snapshot before running the command.

For complete syntax information on the **forcerpoffline** command, see its man page in *Technical Reference: RSCT for AIX* or *Technical Reference: RSCT for Multiplatforms*.

Taking a peer domain or some of its nodes offline in a CAA environment

Because the peer domain corresponding to a CAA cluster remains defined and online for the life of the cluster, there is no notion of taking individual nodes, or the entire cluster, offline.

The **stoprpdomain** and **stoprpnode** commands succeed, but have no effect.

Removing individual nodes from, or removing an entire, peer domain

When upgrading hardware or otherwise reorganizing your peer domain configuration, you may need to remove individual nodes from a peer domain, or else remove an entire peer domain definition.

To remove individual nodes from a peer domain or an entire peer domain:

- Use the **rmrpnode** command to remove a node from a peer domain.
- Use the **rmrpdomain** command to remove a peer domain definition.

Removing a node from a peer domain

In order to remove a node from a peer domain, the node must be offline.

If the node you wish to remove is not currently offline, you must use the **stoprpnode** command to take it offline. For more information on the **stoprpnode** command, see "Taking a peer domain node offline" on page 64.

To see if the node is offline, issue the **lsrpnode** command from any node in the peer domain.

```
lsrpnode
```

Issuing this command lists information about the nodes defined in the peer domain. For example:

```
Name                OpState             RSCTVersion
nodeA               offline             3.1.5.0
nodeB               online              3.1.5.0
nodeC               online              3.1.5.0
nodeD               online              3.1.5.0
nodeE               offline             3.1.5.0
```

In this example, *nodeA* and *nodeE* are offline and can be removed.

To remove a node, issue the **rmrpnode** command from any online node in the peer domain, passing the **rmrpnode** command the peer domain node name of the node to remove. For example, to remove *nodeA*:

```
rmrpnode nodeA
```

You can also remove multiple nodes from the peer domain:

```
rmrpnode nodeA nodeE
```

Since removing a node changes the domain configuration definition, the **rmrpnode** command, by default, requires a configuration quorum. The configuration quorum for this command is either a majority of nodes or exactly half the nodes provided the configuration resource manager can remove the configuration from at least one of the offline nodes. You can override the need for a configuration quorum and force node removal by specifying the **-f** option on the **rmrpnode** command. For example:

```
rmrpnode -f nodeA
```

Without the **-f** option, attempting to remove the last quorum node or preferred node will fail. With the **-f** option, attempting to remove the last quorum node or preferred node will result in all other nodes being converted to quorum nodes or preferred nodes (respectively).

For complete syntax information on the **rmrpnode** and **lsrpnode** commands, see their man pages in *Technical Reference: RSCT for AIX* or *Technical Reference: RSCT for Multiplatforms*.

Removing a peer domain

Removing a peer domain involves removing the peer domain definition from each node on the peer domain.

You can remove the peer domain definition by issuing the **rmrpdomain** command from any online node in the peer domain. You pass the **rmrpdomain** command the name of the peer domain. For example, to remove the peer domain *ApplDomain*:

```
rmrpdomain ApplDomain
```

The **rmrpdomain** command removes the peer domain definition on all of the nodes that are reachable from the node where the command was issued. If all the nodes are reachable, then the command will attempt to remove the peer domain definition from all nodes. If a node is not reachable from the node where the **rmrpdomain** is run (for example, the network is down or the node is inoperative), the **rmrpdomain** command will not be able to remove the peer domain definition on that node. If there are

nodes that are not reachable from the node where the **rmrpdomain** command was run, you will need to run the **rmrpdomain** command from each node that did not have their peer domain definition removed. Include the **-f** option to force the removal:

```
rmrpdomain -f ApplDomain
```

You can also use the **-f** flag if an RSCT subsystem (such as Topology Services or Group Services) rejects the **rmrpdomain** command because a peer domain resource is busy. The **-f** flag will force the RSCT subsystems to take the peer domain offline and remove the peer domain definitions regardless of the state of peer domain resources.

For complete syntax information on the **rmrpdomain** command, see its man page in *Technical Reference: RSCT for AIX* or *Technical Reference: RSCT for Multiplatforms*.

Removing a peer domain, or its nodes, in a CAA environment

You can remove a peer domain, or its nodes, in a CAA environment.

Removing a peer domain in a CAA environment proceeds similarly to removing a peer domain that is not running in a CAA environment, through the use of the **rmrpdomain** command. The peer domain and the CAA cluster are both deleted.

Removing a node from a peer domain in a CAA environment proceeds identically to removing a node from a peer domain that is not running in a CAA environment, through the use of the **rmrpnode** command. The node is removed from the peer domain and from the CAA cluster.

Changing a peer domain's quorum type

A peer domain's quorum type is used to calculate startup quorum, configuration quorum, and operational quorum.

As described in “Quorum types” on page 39, a peer domain's quorum type is used to calculate startup quorum, configuration quorum, and operational quorum. The peer domain's quorum type will either be the default for your environment, or one you explicitly specify. When creating a peer domain, you can specify the quorum type using the **mkrpdomain** command's **-Q** flag.

Once a peer domain is created, you can also modify its quorum type using the generic RMC command **chrsrc**. You can use the **chrsrc** command to modify the QuorumType attribute of the PeerNode class.

For example, to modify a peer domain's quorum type to quick startup mode, you would enter the following command from a node that is online in the peer domain.

```
chrsrc -c IBM.PeerNode QuorumType=1
```

For detailed syntax information on the **chrsrc** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Changing a peer domain's quorum nodes

You can use the **chrsrc** command to change the value of the **IsQuorumNode** attribute for a node.

For instance, the following command changes node *nodeA* to a quorum node:

```
export CT_MANAGEMENT_SCOPE=2;chrsrc -s "Name == 'nodeA'" IBM.PeerNode IsQuorumNode=1
```

Conversely, the following command changes node 'nodeA' to not being a quorum node:

```
export CT_MANAGEMENT_SCOPE=2;chrsrc -s "Name == 'nodeA'" IBM.PeerNode IsQuorumNode=0
```

Changing IP addresses in a peer domain

The configuration resource manager automatically monitors for such configuration changes as IP address updates in an RSCT peer domain. When such changes are detected, the configuration resource manager updates the online peer domain configuration to keep the configuration synchronized across all nodes of the peer domain. Because IP addresses are the critical path to a node, there are some rules to follow when updating IP addresses so that the nodes in a peer domain can continue to be accessed by the configuration resource manager and other cluster subsystems.

Table 18 outlines the rules for changing IP addresses in a peer domain.

Table 18. Rules for changing IP addresses in a peer domain

If a node has...	Then...
Multiple IP addresses communicating with all the nodes in the peer domain, and you want to change a subset of these addresses	There are no restrictions to changing IP addresses.

Multiple IP addresses communicating with all the nodes in the peer domain, and you want to change all of these addresses	You must not change all of the IP addresses at the same time. Leave at least one IP address unchanged so that communication to the node will not be lost. If communication to a node is lost, the other nodes in the domain will consider the changed node to be offline since they only know it by its old IP address. In addition, the configuration resource manager on the changed node will have no way of telling the remaining nodes about the change.
	
	To change IP addresses, you can do so either by changing the IP addresses one at a time or by changing all but one in a single request. Once the node has been harvested after the first change and the cluster configuration is updated with the change, you can then proceed to modify the next or the last unchanged IP address.
	
	The configuration resource manager checks for changes periodically (every minute or so) and applies any detected changes to the cluster configuration. After making a change, wait about 1 minute and 30 seconds for the change to be reflected or until the following command reflects the change:
	`lsrsrc IBM.NetworkInterface`
	
	Alternatively, you can force the configuration resource manager to detect the change by running the following command on the node where the IP address was changed:
	`refrsrc IBM.NetworkInterface`

A single IP address communicating with all of the nodes in the peer domain	1. Remove the node from the peer domain (using the **rmrpnode** command, as described in “Removing a node from a peer domain” on page 66). 2. Change its IP address. 3. Add the node back to the peer domain. (Using the **addrpnode** command, as described in “Adding nodes to an existing peer domain” on page 60).

Table 18. Rules for changing IP addresses in a peer domain (continued)

If a node has...	Then...
IPv4 and IPv6 addresses, and you want the IPv6 addresses to be visible as resources in the **IBM.NetworkInterface** class along with the IPv4 addresses, and to be used for heartbeating and internal peer domain operations	Do one of the following: 1. Issue the **mkrpdomain -6** command. 2. Set the value of the **IPv6Support** persistent class attribute in the **IBM.NetworkInterface** resource class to **1** (True) using the **chrsrc -c** command. The **IPv6Support** attribute controls the ability to select IPv6 entries in **IBM.NetworkInterface**. The default value of this attribute is **0** (False), which indicates that IPv6 addresses are not enabled in a peer domain. **Note:** Even when the **IPv6Support** persistent class attribute is changed, the currently registered applications receive notifications about the resource addition or deletion only after the **IBM.ConfigRM** resource class is restarted. Multiple **IBM.NetworkInterface** instances can exist for the same network interface, as long as the network interface has an IPv4 address and an IPv6 address. If the interface has multiple IPv6 addresses configured, only one is chosen for representation as a resource. If a global IPv6 address is configured on an interface, it is preferred for representation to the interface's IPv6 link-local address. To identify the type of address that is contained in a given resource, use the **IPVersion** persistent attribute of the **IBM.NetworkInterface** resource class.

An address on an interface represented as a resource, and you would like a different address on the interface in the same address family to be represented as a resource instead	Use the **chrsrc** command to modify the **IPAddress** attribute of the resource. For example, suppose IPv6 global addresses **2009::1** and **2009::2** are configured on **en0**, and **2009::1** is currently represented as an **IBM.NetworkInterface** resource. To represent **2009::2** as a resource instead, run this command while the node on which the interface resides is online in the peer domain:
	`CT_MANAGEMENT_SCOPE=2 chrsrc -s "IPAddress='2009::1'" \`
	`IBM.NetworkInterface "IPAddress=2009::2"`

Changing the network interface harvest interval

The configuration resource manager initiates an internal harvest operation to gather network interface information, by default, every sixty seconds.

Such frequent initiation of the internal harvest operation is of minor consequence when there is little use of system resources and the network configuration does not change, especially in small clusters. However, as the number of nodes increases, frequent harvest operations can cause unnecessary increases in CPU usage and network traffic. Also, in large cluster environments, when the network interface information seldom changes, it may be unnecessary to initiate harvest every minute. Tuning the harvest interval allows you to modify the number of seconds between harvests, controlling how often the system may be taxed to perform a harvest operation.

You can change the harvest interval by using the **chrsrc** command to specify the new interval in seconds. The maximum interval value is 4 hours or 14400 seconds and the minimum harvest interval is 5 seconds. For example, to change the harvest interval to 20 seconds, you can enter the following command on a node:

```
CT_MANAGEMENT_SCOPE=2 chrsrc -c IBM.NetworkInterface HarvestInterval=20
```

To initiate a harvest operation manually, run this command:

```
refrsrc IBM.NetworkInterface
```

Setting the harvest interval to 4 hours also implies that it will take 4 hours for the configuration resource manager to automatically process the change in the adapter configuration.

For more information about the **chrsrc** and **refrsrc** commands, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

You can set the value of the harvest interval attribute either when a single node is offline in independent workstation (IW) mode only, or when a node is online in a peer domain. The RSCT registry structure allows you to set different intervals for each domain to which a given node might belong.

Updating the harvest interval in an online domain requires the use of group protocols to ensure correct adoption of the new interval by all cluster nodes. Such a request is redirected to the configuration resource manager's group leader node. All of the nodes involved must be running RSCT 2.5.0.0 or later on AIX 6.1 or RSCT 3.1.0.0 or later on AIX 7.1. After the new interval is accepted and committed by all nodes, subsequent harvest operations are scheduled by using the new interval value.

Working with communication groups

Several optional tasks are available when working with communication groups.

You can perform the following tasks to work with communication groups:

- List information about the communication groups in a peer domain.
- Modify the characteristics of a communication group.
- When on the rare occasion it might be necessary, manually configure communication groups in a peer domain.

"Understanding communication groups" describes concepts you need to understand before attempting any of the these tasks.

Related tasks:

"Adding nodes to the peer domain" on page 62
Use the **addrpnode** command to add one or more nodes to an RSCT peer domain definition, passing it the IP address or DNS name of the node you want to add.

Understanding communication groups

Communication groups control how liveness checks (in other words, Topology Services' *heartbeats*) are performed between the communication resources within the peer domain.

Each communication group corresponds to a Topology Services heartbeat ring. It identifies the attributes that control the liveness checks between the set of network interfaces and other devices in the group.

The configuration resource manager automatically forms communication groups when a new peer domain is formed using the **mkrpdomain** command. When you bring a peer domain online using the **startrpdomain** command, the configuration resource manager will supply the communication group definition to Topology Services, which creates the actual heartbeat rings needed to perform liveness checks for the peer domain nodes. The configuration resource manager may also form new communication groups as new nodes are added to the peer domain using the **addrpnode** command. When these added nodes are brought online using the **startrpnode** command, the configuration resource manager supplies the modified information to Topology Services, which might modify existing heartbeat rings or create additional heartbeat rings.

With the introduction of support for disk heartbeating, the type of interfaces that make up a communication group needs to be indicated. The persistent resource attribute **MediaType** is added to **IBM.CommunicationGroup** to support this. You can set this using the **mkcomg -M** *media-type* command. Valid values are:

0 Indicates that a communication group consists of interface resources other than IP or disk.

1 Indicates that a communication group consists of IPv4 or IPv6 interface resources.

If the media type is not set using the **mkcomg -M** *media-type* command, this is the default.

2 Indicates that a communication group consists of disk interface resources.

For more information about disk heartbeating, see "Configuring and using disk heartbeat interfaces" on page 78.

The configuration resource manager's automatic creation of communication groups is based on subnet and intersubnet accessibility within address families. For each communication group, the goal is to define a set of interfaces within an address family - that is, IPv4 (AF_INET) or IPv6 (AF_INET6) - with no more than one such interface from each node, each having end-to-end connectivity with the others. Given the restriction that at most one adapter from each node can belong to a given communication group:

- All interfaces in the same subnet will be in the same communication group, unless one node has multiple interfaces in the same subnet.
- Interfaces in different subnets that can communicate with each other can be in the same communication group if they have connectivity.

The configuration resource manager allows you to create your own communication groups and to change the adapter membership in an existing communication group. However, because the configuration resource manager creates the communication groups automatically, such manual configuration is neither necessary nor advisable. *Manual configuration should only be done in unavoidable situations*, such as when a network configuration is too complex for the configuration resource manager's automatic communication group creation algorithm. Manual configuration changes that do not conform to the rules and restrictions discussed in this section could cause partitioning of the peer domain. For more information, see "Manually configuring communication groups" on page 75.

When the configuration resource manager creates communication groups automatically, it gives them such default characteristics as:

Sensitivity
the number of missed heartbeats that constitute a failure.

Period the number of seconds between the heartbeats.

Priority
the importance of this communication group with respect to others.

Broadcast / No Broadcast
whether or not to broadcast (if the underlying network supports it).

Enable/Disable Source Routing
In case of adapter failure, whether or not source routing should be used (if the underlying network supports it).

You can modify a communication group's characteristics using the **chcomg** command, as described in "Modifying a communication group's characteristics" on page 73.

Communication groups in a CAA environment:

In a Cluster-Aware AIX (CAA) environment, the liveness checking configuration, which CAA maintains and uses, is effectively opaque to RSCT.

Thus, communication groups, which define the liveness checking configuration in peer domains that are not operating in a CAA environment, are irrelevant, so are therefore not defined in a peer domain that is running in a CAA environment.

Listing communication groups

The **lscomg** command lists information about the communication groups in a peer domain.

The **lscomg** command lists the following information about the communication groups in a peer domain:

Field Description

Name The name of the communication group

Sensitivity
The sensitivity setting (the number of missed heartbeats that constitute a failure)

Period The period setting (the number of seconds between heartbeats)

Priority
The priority setting (the relative priority of the communication group)

Broadcast
Whether or not broadcast should be used if it is supported by the underlying media

SourceRouting
Whether or not source routing should be used if it is supported by the underlying media

NIMPath
The path to the Network Interface Module (NIM) that supports the adapter types in the communication group

NIMParameters
The NIM start parameters

Grace The number of seconds for the grace period

MediaType
The type of interfaces that make up this communication group

To list general information about the peer domain *ApplDomain*, for example, enter the following command from a node that is online to *ApplDomain*:

```
lscomg
```

The configuration resource manager lists information about the communication groups defined in the peer domain:

```
Name Sensitivity Period Priority Broadcast SourceRouting NIMPath   NIMParameters Grace MediaType
CG1  2           2      1        no        yes           /.../nim  -l 5          0.5   1
```

If there are multiple communication groups defined on the node, and you want only a particular one listed, specify the name of the communication group on the **lscomg** command. For example, to list information about a communication group called **ComGrp**, enter:

```
lscomg ComGrp
```

Interface Resources

Use the **-i** option to display information about the interface resources that refer to the communication group.

For IP communication groups (**MediaType** = **1**), **lscomg -i** displays the following information:

Field Description

Name The peer domain node name of the interface resource that refers to this communication group

NodeName
The host name of the interface resource that refers to this communication group

IPAddress
The IP address of the resource interface that refers to this communication group

SubnetMask
The subnet mask of the resource interface that refers to this communication group

Subnet
The subnet of the resource interface that refers to this communication group

For disk heartbeating (**MediaType** = **2**) and other non-IP types of communication groups (**MediaType** = **0**), **lscomg -i** displays the following information:

Field **Description**

Name The peer domain node name of the interface resource that refers to this communication group

NodeName
The host name of the interface resource that refers to this communication group

DeviceInfo
Information about the device

MediaType
The type of interfaces that make up this communication group

To list interface resource information for a communication group, use the **lscomg -i** command:

```
lscomg -i ComGrp1
```

If **ComGrp1** is an IP communication group (**MediaType** = **1**), the output will look like this:

```
Name   NodeName     IPAddr       SubnetMask     Subnet
eth0   n24.ibm.com  9.234.32.45  255.255.255.2  9.235.345.34
eth0   n25.ibm.com  9.234.32.46  255.255.255.2  9.235.345.34
```

If **ComGrp1** is a disk heartbeating (**MediaType** = **2**) or other non-IP type of communication group (**MediaType** = **0**), the output will look like this:

```
Name   NodeName     DeviceInfo    MediaType
eth0   n24.ibm.com  /dev/hdisk0   2
eth0   n25.ibm.com  /dev/hdisk1   2
```

If you want to change any of the settings of a communication group, you can use the **chcomg** command, as described in "Modifying a communication group's characteristics."

Modifying a communication group's characteristics

A communication group has a number of properties or characteristics that determine its behavior.

These properties are established when the communication group is created and include such tunables as the group's sensitivity, period, and priority settings. Using the **chcomg** command, you can change the settings, and so the behavior, of a communication group. To see the current settings for a communication group, use the **lscomg** command as described in "Listing communication groups" on page 71.

You can also use the **chcomg** command to modify a communication group's network interface assignment. You typically do not need to modify this, and in fact should perform such manual configuration only in unavoidable situations. See "Modifying a communication group's network interface" on page 75 for more information.

Since the **chcomg** command modifies the domain configuration definition, it will not change a communication group's characteristics unless a majority of nodes are online in the domain. If such a *configuration quorum* exists, the domain configuration definition can be modified.

For complete syntax information on the **chcomg** command, see its man page in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying a communication group's sensitivity setting:

A communication group's sensitivity setting refers to the number of missed Topology Services' heartbeats that constitute a failure.

1. To determine what a communication group's sensitivity setting is, use the **lscomg** command as described in "Listing communication groups" on page 71.
2. To modify a communication group's sensitivity setting, use the **chcomg** command with its **-s** flag. The sensitivity setting must be an integer greater than or equal to 2.

Example: To modify the communication group *ComGrp1* so that its sensitivity setting is 4, issue the following command on a node that is online in the peer domain:

```
chcomg -s 4 ComGrp1
```

Modifying a communication group's period setting:

A communication group's period setting refers to the number of seconds between Topology Service's heartbeats.

1. To determine what a communication group's period setting is, use the **lscomg** command as described in "Listing communication groups" on page 71.
2. To modify a communication group's period setting, use the **chcomg** command with its **-p** flag. The period setting must be an integer greater than or equal to 1.

Example: To modify the communication group *ComGrp1* so that its period is 3, issue the following command on a node that is online in the peer domain.

```
chcomg -p 3 ComGrp1
```

Modifying a communication group's priority setting:

A communication group's priority setting refers to the importance of this communication group with respect to others and is used to order the Topology Services heartbeat rings. The lower the number means the higher the priority. The highest priority is 1.

1. To determine what a communication group's priority setting is, use the **lscomg** command as described in "Listing communication groups" on page 71.
2. To modify a communication group's priority setting, use the **chcomg** command with its **-t** flag.

Example: To modify the communication group *ComGrp1* so that its priority is 3, issue the following command on a node that is online in the peer domain:

```
chcomg -t 3 ComGrp1
```

Modifying a communication group's broadcast setting:

A communication group's broadcast setting specifies whether or not broadcast will be used (provided the underlying network supports it).

1. To determine what a communication group's broadcast setting is, use the **lscomg** command as described in "Listing communication groups" on page 71.
2. To modify a communication group's broadcast setting so that broadcast operations are enabled, use the **chcomg** command with its **-b** flag.
3. Alternately, to modify a communication group's broadcast setting so that broadcast operations are disabled, use the **chcomg** command with its **-x b** flag.

For example:

- To modify the communication group *ComGrp1* so that broadcast will be used (provided the underlying network supports it), issue the following command on a node that is online in the peer domain.

```
chcomg -b ComGrp1
```

- To modify the communication group *ComGrp1* so that broadcast will **not** be used, issue the following command on a node that is online in the peer domain.

```
chcomg -x b ComGrp1
```

Modifying a communication group's source routing setting:

A communication group's source routing setting specifies whether or not source routing will be used in case of adapter failure (provided the underlying network supports it).

To determine what a communication group's source routing setting is, use the **lscomg** command as described in "Listing communication groups" on page 71.

By default, source routing is enabled. To modify a communication group's broadcast setting so that source routing is disabled, use the **chcomg** command with its **-x r** flag. For example, to modify the communication group **ComGrp1** so that source routing will not be used, issue the following command on a node that is online in the peer domain.

```
chcomg -x r ComGrp1
```

To modify a communication group's source routing setting so that source routing is enabled, use the **chcomg** command with its **-r** flag. For example, to modify the communication group *ComGrp1* so that source routing will be used in case of adapter failure, issue the following command on a node that is online in the peer domain.

```
chcomg -r ComGrp1
```

Manually configuring communication groups

You can change the adapter membership of an existing communication group, create a new communication group, and remove communication groups. However, be aware that, under normal circumstances, ***manual configuration is unnecessary and unadvisable***.

Normally, communication groups are automatically created when a new peer domain is formed by the **mkrpdomain** command, and modified when a node is added by the **addrpnode** command. When the peer domain is brought online by the **startrpdomain** command or the new node is brought online by the **startrpnode** command, the configuration resource manager supplies the communication group information to Topology Services which will create/modify the heartbeat rings.

Manual configuration may be exercised, but ***only in unavoidable situations*** (such as when a network configuration is more complex than our automatic communication algorithm has anticipated or can handle).

Note: The three configuration commands—**chcomg**,**mkcomg**, and **rmcomg**—all modify a domain's configuration definition and, for that reason, will not make any changes unless a majority of nodes are online in the domain. If such a *configuration quorum* exists, the domain configuration definition can be modified.

Modifying a communication group's network interface:

You can also use the **chcomg** command to modify a communication group's network interface assignment.

Additionally, "Modifying a communication group's characteristics" on page 73 describes how to use the **chcomg** command to modify a communication group's tunables (such as its sensitivity, period, and priority settings). We do not recommend you do this, and any changes you make must conform to the following rules. These are the same rules that the configuration resource manager uses in creating communication groups automatically. Failure to follow these rules may cause partitioning of the peer domain. The rules are:

- At most, one adapter from each node can belong to a given communication group.
- All adapters in the same subnet will be in the same communication group.
- Adapters on different subnets that can communicate with each other may be in the same communication group.

In addition, because RSCT uses IP broadcast to optimize its communication, the following rules should be followed when configuring network interfaces.

- For each network interface, its broadcast address or subnet mask should be consistent with each other. That is: **Bcast address = IP address OR (negated netmask)**. For example, if IP address is 1.2.3.4 and netmask is 255.255.255.0, then the broadcast address should be 1.2.3.255.
- The subnet mask and broadcast addresses should be the same across all the interfaces that belong to the same subnet. Interfaces that belong to different subnets are allowed to have different subnet masks.

Use the **chcomg** command modify a communication group's network interface, as follows:

- Assign the communication group to a network interface using either the **-i** flag or the **-S** flag with the **n** clause.
  - Using the **-i** flag and **n** clause, you can assign the communication group to the network interface by specifying the network interface name and, optionally, the name of the node where the resource can be found.
  - Using the **-S** flag with the **n** clause, you can assign the communication group to the network interface by specifying a selection string.
- If necessary, use the **-e** flag to specify the path to the Network Interface Module (NIM) that supports the adapter type, and the **-m** flag to specify any character strings you want passed to the NIM as start parameters. It is likely that the NIM path (which is */usr/sbin/rsct/bin/hats_nim*) is already specified in the communication group definition; issue the **lscomg** command as described in "Listing communication groups" on page 71 to ascertain this.

With the introduction of IPv6 support in heartbeating, using the **-i n:** flag or the **-S n:** flag to modify the network interfaces in a communication group in this way must specify whether the IPv4 or IPv6 addresses configured on the specified interfaces and represented as resources should be affected. This is required because IPv4 and IPv6 addresses do not have connectivity between each other, so placing them in the same communication group would not make sense. To address this issue, the **-6** flag is introduced for the **-i n:** and **-S n:** flags. If **-6** is not supplied, only IPv4 addresses represented as resources on the specified interfaces will have their communication group changed to the one specified in the command. Conversely, if **-6** is supplied, only IPv6 address interface resources will be so changed. For example, the command:

```
chcomg -6 -i n:en0:nodeA,en0:nodeB newCG
```

would move the IPv6 addresses configured on **en0** on **nodeA** and **nodeB** and represented as interface resources to the **newCG**communication group, but leave the IPv4 addresses represented as resources on those interfaces in their current communication group.

To modify the **ComGrp1** communication group's network interface to the network interface resource named **eth0** on **nodeB**, you would enter the following from a node that is online in the peer domain.

```
chcomg -i n:eth0:nodeB ComGrp1
```

To specify the NIM path and options (in this case, the option is **"-l 5"** to set the logging level), you would enter the following from a node that is online in the peer domain.

```
chcomg -i n:eth0:nodeB -e /usr/sbin/rsct/bin/hats_nim -m "-l 5" ComGrp1
```

To assign the communication group *ComGrp1* to the network interface resource that uses the subnet 9.123.45.678, you would enter the following from a node that is online in the peer domain.

```
chcomg -S n:"Subnet==9.123.45.678" ComGrp1
```

Creating a communication group:

Under normal circumstances, the configuration resource manager creates communication groups automatically when a new peer domain is formed, and modifies them as new nodes are added to the peer domain. You should not need to create your own communication groups.

This ability is provided only to address special situations such as when a network configuration is more complex than our automatic communication group algorithm has anticipated or can handle.

Use the **mkcomg** command to create a communication group. One of the key things that you will need to specify is the communication group's network interface assignment. When making such assignments, you must conform to the following rules. These are the same rules that the configuration resource manager uses when creating communication groups automatically. Failure to follow these rules may cause partitioning of the peer domain. The rules are:

- At most, one adapter from each node can belong to a given communication group.
- All adapters in the same subnet will be in the same communication group.
- Adapters on different subnets that can communicate with each other may be in the same communication group.

To set a communication group's network interface:

- Assign the communication group to a network interface using either the **-i** flag or the **-S** flag with the **n** clause.
  - Using the **-i** flag and n clause, you can assign the communication group to the network interface by specifying the network interface name and, optionally, the name of the node where the resource can be found.
  - Using the **-S** flag with the **n** clause, you can assign the communication group to the network interface by specifying a selection string.
- Use the **-e** flag to specify the path to the Network Interface Module (NIM). In RSCT, a NIM is a process started by the Topology Services' daemon to monitor a local adapter. The NIM executable is located at */usr/sbin/rsct/bin/hats_nim*, and one instance of the NIM process exists for each local adapter that is part of the peer domain. In addition to the **-e** flag, you can use the **-m** flag to specify any character strings you want passed to the NIM as start parameters

As with using **chcomg** to modify the communication group for a set of interfaces, with the introduction of IPv6 support, **mkcomg** also allows you to specify whether IPv4 or IPv6 address interface resources should be affected by its operation. A **-6** flag is introduced that performs this function in the same way as for **chcomg**.

To create the communication group *ComGrp1*, specifying the network interface resource name *eth0* on *nodeB*, you would enter the following from a node that is online in the peer domain:

```
mkcomg -i n:eth0:nodeB -e /usr/sbin/rsct/bin/hats_nim -m "-l 5" ComGrp1
```

The NIM parameters in the preceding example (`-l 5`) set the logging level.

Example: To create the communication group *ComGrp1*, specifying the network interface resource that uses the subnet 9.123.45.678, you would enter the following from a node that is online in the peer domain:

```
mkcomg -S n:"Subnet == 9.123.45.678" -e /usr/sbin/rsct/bin/hats_nim
-m "-l 5" ComGrp1
```

You can also set a number of tunables for the Topology Services' heartbeat ring when issuing the **mkcomg** command. You can specify the:

- Sensitivity setting (the number of missed heartbeats that constitute a failure) using the **-S** flag.
- Period setting (the number of seconds between the heartbeats) using the **-p** flag.

- Priority setting (the importance of this communication group with respect to others) using the **-t** flag.
- Broadcast setting (whether or not to broadcast if the underlying network supports it) using the **-b** (broadcast) or **-x b** (do not broadcast) flags.
- Source routing setting (in case of adapter failure, whether or not source routing should be used if the underlying network supports it) using the **-r** (use source routing) or **-x r** (do not use source routing) flags.

The following command creates the *ComGrp1* communication group as before, but also specifies that:
- its sensitivity is 4
- its period is 3
- its priority is 2
- broadcast should be used
- source routing should not be used

```
mkcomg -s 4 -p 3 -t 2 -b -x r -i n:eth0:nodeB -e /usr/sbin/rsct/bin/hats_nim
-m "-l 5" ComGrp1
```

You can display all of the settings for a communication group using the **lscomg** command (as described in "Listing communication groups" on page 71). To change any of the settings, you can use the **chcomg** command (as described in "Modifying a communication group's characteristics" on page 73). For complete syntax information on the **mkcomg** command, see its man page in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Removing a communication group:

The **rmcomg** command enables you to remove an already-defined communication group definition from a peer domain. As with all the manual configuration commands for communication groups, you will not normally need to do this. Manual configuration must be exercised with caution and only in unavoidable situations.

To list the communication groups in the peer domain, you can use the **lscomg** command as described in "Listing communication groups" on page 71. Before removing a communication group, you must first use the **chcomg** command to remove interface resource references to the communication group (as described in "Modifying a communication group's network interface" on page 75).

To remove a communication group, supply its name to the **rmcomg** command.

Example: To remove the communication group *ComGrp1*, issue the following command from a node that is online in the peer domain:

```
rmcomg ComGrp1
```

For complete syntax information on the **rmcomg** command, see its online man page, the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Configuring and using disk heartbeat interfaces

On AIX 6.1, AIX 7.1, and Linux, the configuration resource manager permits the configuring of the Topology Services subsystem to "heartbeat" through disk media. This type of configuration, which is referred to as *disk heartbeating*, also permits the Group Services subsystem to perform reliable messaging through the same disk media.

Configuring disk heartbeating results in a node heartbeating and communication path that contrasts with, and is independent of, IP heartbeating and reliable messaging through network interfaces. A single disk heartbeating configuration is made up of:
- A communication group with a **MediaType** attribute of **2** (**Disk**)

- A pair of heartbeat interface devices that refer to separate nodes (**NodeNameList** attribute), to the same communication group (**CommGroup** attribute), and to the same disk that is shared between the nodes (**DeviceInfo** attribute). This disk is not referred to by any other heartbeat interface on the same node.

Verifying disk configuration

To verify disk configuration on AIX, enter:

```
lspv
```

To verify disk configuration on Linux, use one of the following commands.

To display physical volume IDs (PVIDs), enter:

```
pvdisplay
```

To display logical volume IDs (LVIDs), enter:

```
lvdisplay
```

To display multipath worldwide IDs (WWIDs), enter:

```
multipath -l
```

Configuring heartbeat interfaces

A heartbeat interface's **DeviceInfo** attribute value must conform to one of these valid formats:

ID type
: **Pattern**

logical volume
: LVID=* | LVID:*

multipath
: MPATH=* | MPATH:

physical volume
: PVID=* | PVID:*

raw disk
: /dev/hdisk*

RDAC worldwide name
: WWN=* | WWN:*

The mapping between **MediaType** attribute enumeration values and their respective string values follows:

Enumeration value	String value
0	UserDefined
1	IP
2	Disk

The **lscomg** command displays these values in this format:

```
0 (UserDefined)
1 (IP)
2 (Disk)
```

Only the enumeration value is accepted by external interface commands.

Heartbeat interfaces can be created (using **mkrsrc**), changed (using **chrsrc**), examined (using **lsrsrc**), and removed (using **rmrsrc**). For all commands that are related to disk heartbeating tasks, as shown in the sections that follow:

- Attributes that are not shown for a command are not valid for that command
- All attributes are selectable (that is, they can be specified using the **-s** flag with commands that include this flag)
- When present in the tables that follow, the **Force** column indicates which attributes are affected by the **Force** option
- **Force=0** and the absence of the **Force** option are equivalent
- An error is reported if restrictions are not met

Creating heartbeat interface resources

Use the **mkrsrc** command to create a heartbeat interface resource. For example, enter:

```
CT_MANAGEMENT_SCOPE=2 mkrsrc IBM.HeartbeatInterface attributes [Force=0|1]
```

Table 19. Attributes specified when creating heartbeat interface resources

Name	Required?	Type	Default value	Force	Restrictions
Name	yes	string			No blanks, 36 bytes or less
DeviceInfo	yes	string		0 1	• Must exist • Must have a valid ID pattern • Must be **MediaType==2** only
CommGroup	yes	string	1	0 1	• Must exist • Created in **IBM.CommunicationGroup** if it doesn't exist
Qualifier	no	string			
NodeNameList	yes	array			Must consist of 1 or 2 peer nodes (for example: **"{'node1', 'node2'}"**
MediaType	yes	uint32	0		0 (user defined), 2 (disk)

- Specifying two nodes leads to the creation of two interfaces, each with one node
- Each **Name** attribute on a node must be unique
- Each **DeviceInfo** attribute on a node must be unique
- Different **DeviceInfo** attributes on the same node cannot be in the same communication groups
- There is a maximum of two names (**Name** attribute) per communication group (**CommGroup** attribute)
- **Force=1** does not override the other restrictions in this list

Changing heartbeat interface resources

- Each **Name** attribute on a node must be unique
- Each **DeviceInfo** attribute on a node must be unique
- Different **DeviceInfo** attributes on the same node cannot be in the same communication groups
- There is a maximum of two names (**Name** attribute) per communication group (**CommGroup** attribute)
- Any resulting unreferenced communication groups (**CommGroup** attribute) in **IBM.CommunicationGroup** are not removed

Use the **chrsrc** command to change a heartbeat interface resource. For example, enter:

```
CT_MANAGEMENT_SCOPE=2 chrsrc -s "selection criteria" IBM.HeartbeatInterface attributes
```

Table 20. Attributes specified when changing heartbeat interface resources

Name	Type	Restrictions
Name	string	No blanks, 36 bytes or less
DeviceInfo	string	• Must exist • Must have a valid ID pattern • Must be **MediaType==2** only
CommGroup	string	Must exist
Qualifier	string	
NodeNameList	array	ReadOnly
MediaType	uint32	ReadOnly

Examining heartbeat interface resources

Use the **lsrsrc** command to examine heartbeat interface resources. For example, enter:

```
CT_MANAGEMENT_SCOPE=2 lsrsrc IBM.HeartbeatInterface
```

Removing heartbeat interface resources

Use the **rmrsrc** command to remove a heartbeat interface resource. For example, enter:

```
CT_MANAGEMENT_SCOPE=2 rmrsrc -s "selection_criteria" IBM.HeartbeatInterface [Force=0|1]
```

- Any resulting unreferenced communication groups in **IBM.CommunicationGroup** are not removed

Configuring communication groups

Communication groups can be created (using **mkrsrc** or **mkcomg**), changed (using **chrsrc** or **chcomg**), examined (using **lsrsrc** or **lscomg**), or removed (using **rmrsrc** or **rmcomg**).

For all commands that are used when configuring communication groups related to disk heartbeating, as shown in the sections that follow:

- Attributes that are not shown have no specific relevance to disk heartbeating
- All attributes are selectable (that is, they can be specified using the **-s** flag with commands that include this flag)
- An error is reported if restrictions are not met

Creating communication group resources

Use the **mkcomg** command or the **mkrsrc** command to create a communication group resource.

To create a communication group resource using the **mkcomg** command, enter:

```
mkcomg [options] CG_name
```

- Use the **mkcomg -M** *media-type* command to specify the type of interfaces that make up a communication group. If the **-M** flag is not specified, the default value is **1** (IP).
- The **-i** and **-S** flags permit changing a heartbeat interface's **CommGroup** attribute when creating a communication group. For example:

  ```
  -i h:HBI_name[:node_name],HBI_name[:node_name],... | -S h:"HBI_selection_string"
  ```

  Note the **h:** prefix for heartbeat interfaces, which is similar to the **n:** prefix for network interfaces.

To create a communication group resource using the **mkrsrc** command, enter:

```
CT_MANAGEMENT_SCOPE=2 mkrsrc IBM.CommunicationGroup attributes
```

Table 21. Attributes specified when creating communication group resources

Name	Required?	Type	Default value	Restrictions
Period	no	uint32	1 for **MediaType==1** (IP) 2 for others	
PingGracePeriodMilliSec	no	uint32		Valid only for **MediaType==1** (IP)
MediaType	no	uint32	0	0 (user defined), 1 (IP), 2 (disk)

Changing communication group resources

Use the **chcomg** command or the **chrsrc** command to change a communication group resource.

To change a communication group resource using the **chcomg** command, enter:

```
chcomg [options] CG_name
```

The **-i** and **-S** flags permit changing a heartbeat interface's **CommGroup** attribute when changing a communication group. For example:

```
-i h:HBI_name[:node_name],HBI_name[:node_name],... | -S h:"HBI_selection_string"
```

Note the **h:** prefix for heartbeat interfaces, which is similar to the **n:** prefix for network interfaces.

To change a communication group resource using the **chrsrc** command, enter:

```
CT_MANAGEMENT_SCOPE=2 chrsrc -s "selection criteria" IBM.CommunicationGroup attributes
```

Table 22. Attributes specified when changing communication group resources

Name	Type	Restrictions
PingGracePeriodMilliSec	uint32	Valid only for **MediaType==1** (IP)
MediaType	uint32	ReadOnly

Examining communication group resources

Use the **lscomg** command or the **lsrsrc** command to examine communication group resources.

To examine communication group resources using the **lscomg** command, enter:

```
lscomg [options] CG_name
```

To examine communication group resources using the **lsrsrc** command, enter:

```
CT_MANAGEMENT_SCOPE=2 lsrsrc IBM.CommunicationGroup
```

Additional information displayed for disk heartbeating support

- The output of the **lscomg** [**-t**] command (table format, the default) now includes the **MediaType** column.
- The output of the **lscomg -l** command (long format) now includes the **MediaType** field.
- The output of the **lscomg -i** *comm_group* command now includes the **DeviceInfo** and **MediaType** columns when *comm_group* is a disk heartbeating communication group (**MediaType** = **2**). This command obtains associated **DeviceInfo** attributes from **/IBM/HeartbeatInterface/Resources**.

Removing communication group resources

Use the **rmcomg** command or the **rmrsrc** command to remove a communication group resource.

To remove a communication group resource using the **rmcomg** command, enter:

```
rmcomg CG_name
```

To remove a communication group resource using the **rmrsrc** command, enter:

```
CT_MANAGEMENT_SCOPE=2 rmrsrc -s "selection criteria" IBM.CommunicationGroup
```

- Communication groups to which heartbeat interfaces refer are not removed
- There is no Force option for removing communication group resources

Modifying Topology Services and Group Services parameters

You can use the **chrsrc** command to change the control parameters used by Topology Services or Group Services for an online cluster through IBM.RSCTParameters resource class.

For a complete discussion of Topology Services, see "The Topology Services subsystem" on page 286. For a complete discussion of Group Services, see "The Group Services subsystem" on page 308. To obtain more information on the IBM.RSCTParameters resource class, use the **lsrsrcdef** command (as described in "Displaying attribute definition information for a resource or a resource class" on page 135).

An IBM.RSCTParameters resource class instance is created for each cluster when the cluster is first brought online. The control parameters include:

- Topology Services log size (**TSLogSize**)
- fixed priority (**TSFixedPriority**)
- pinned regions (**TSPinnedRegions**)
- Group Services log size (**GSLogSize**)
- maximum directory size (**GSMaxDirSize**)

An instance of the class is created automatically for a cluster when the cluster is brought online the first time. The default values for these parameters will be used when it is created.

To view or change the RSCT parameters, you use generic RMC commands (**lsrsrc** and **chrsrc** as described below). To use these generic RMC commands, you need to first set the management scope to 2.

```
export CT_MANAGEMENT_SCOPE=2
```

This tells RMC that the management scope is a peer domain.

To view the parameter values, issue the command:

```
lsrsrc -c IBM.RSCTParameters
```

These values are tunable. They can be changed using one of the following commands:

```
chrsrc -c IBM.RSCTParameters Attr=Value...
```

For example, to tell Topology Services to ping both code and data regions (a value of 3), execute the following command:

```
chrsrc -c IBM.RSCTParameters TSPinnedRegions=3
```

The command is equivalent to the Topology Services tunable command (**cthatstune**) or the Group Services tunable command (**cthagstune**).

Defining preferred nodes for Group Services name server and group leader selection

Because the Group Services daemons that act in the name server or group leader roles can consume a significant amount of system resources when it performs those duties, it is sometimes desirable to define the preferred nodes for Group Services to consider (or non-preferred nodes to avoid) when selecting candidates for these roles.

By default, when Group Services initially establishes the name server in a domain, it selects the lowest-numbered node as the name server candidate node, and the node of the first provider to create a group as the group leader. During recovery, the default is to choose the next node in the node list as the new name server candidate node, and the next node in the group member list as the new group leader.

Because the Group Services daemons acting in the name server or group leader roles can consume a significant amount of system resources while performing those duties, it is sometimes desirable to define the preferred nodes for Group Services to consider (or non-preferred nodes to avoid) when selecting candidates for these roles.

A *preferred node* for the selection of name server and group leader candidates is indicated by a node that has the optional **IsPreferredGSGL** attribute set to 1. The **IsPreferredGSGL** attribute is in the **IBM.PeerNode** class.

When preferred nodes are defined in a peer domain:

- Group Services select the lowest-numbered, preferred node to be the initial name server candidate node. Likewise, during name server recovery, Group Services select the next node in the node list that is also a preferred node to be the new name server candidate node.
- When a provider first attempts to create a group, if the provider resides on a preferred node, that node becomes the group leader. If the provider does not reside on a preferred node but the name server does, then the name server node becomes the group leader. Otherwise, the provider's node becomes the group leader by default. During group leader recovery, Group Services select the next node in the group member list that is also a preferred node to be the new group leader.

When creating a peer domain, you can use the **mkrpdomain** command's **-f** option with a node definition file to specify which nodes are to be preferred nodes. In the node definition file, node names that are followed by `@P` will be considered as preferred nodes for name server and group leader candidate selection. Node names that are followed by `@!P` are considered as non-preferred nodes.

Similarly, when adding one or more nodes to a peer domain, you can also use the **addrpnode** command's **-f** option with a node definition file to specify which nodes are to be considered preferred nodes in the same manner as for the **mkrpdomain** command.

To display whether the nodes in a peer domain are preferred nodes for name server and group leader candidate selection, use the **lsrponode** command with the **-P** option. For instance:

```
lsrpnode -P
```

The following output is displayed:

```
Name    OpState   RSCTVersion   Preferred
nodeA   Online    3.1.5.0       yes
nodeB   Online    3.1.5.0       yes
nodeC   Online    3.1.5.0       no
```

| You can also directly display the value of the **IsPreferredGSGL** resource attribute of the IBM.PeerNode
| class for a node by using the **lsrsrc** command. For instance:

```
lsrsrc -s 'Name="nodeA"' IBM.PeerNode IsPreferredGSGL
```

| You can use the **chrsrc** command to change the value of the **IsPreferredGSGL** attribute for a node. For instance, the following command changes the node to a preferred node:

```
chrsrc -s 'Name=="nodeA"' IBM.PeerNode IsPreferredGSGL=1
```

Conversely, the following command changes the node to not being a preferred node:

```
chrsrc -s 'Name=="nodeA"' IBM.PeerNode IsPreferredGSGL=0
```

Even when preferred nodes are defined, a non-preferred node can still end up being selected as the name server or group leader if no preferred nodes are available. In this case, the default algorithm is used to select the node for the name server or group leader role. Subsequently, as the topology is refreshed and changes occur to the preferred node list, Group Services are notified of these changes but no immediate action is taken to change name server or group leader nodes. The arrival of a preferred node does not cause the name server or group leader role to switch to that node from a non-preferred node. Such node changes can only occur during recovery of a failing name server or group leader node.

Related information:

"The Group Services subsystem" on page 308
The configuration resource manager uses the Group Services subsystem to provide distributed coordination, messaging, and synchronization among nodes in an RSCT peer domain.

mkrpdomain Command

addrpnode Command

lsrsrc Command

lsrpnode Command

chrsrc Command

Determining how your system responds to domain partitioning and subsystem daemon failure

There are various ways that you can configure your peer domain to determine how the configuration resource manager calculates operational quorum and responds to domain partitioning and subsystem daemon failure.

To protect data, the configuration manager uses a quorum of nodes (called an *operational quorum*) to determine whether resources can be safely activated without creating conflicts with other subsystems. For more information, see "Operational quorum" on page 38.

There are various tasks that you can configure your peer domain to determine how the configuration resource manager calculates operational quorum and responds to domain partitioning and subsystem daemon failure. These tasks are all performed by issuing standard Resource Management and Control (RMC) commands, such as **lsrsrc** and **chrsrc**, to set attributes of various resources of the configuration resource manager. For this reason, it is important that you first understand RMC and how, along with the various resource managers, it enables you to manage the resources of your system in a consistent and generic manner. See "Monitoring resources using RMC and resource managers" on page 126 for more information.

You can:

- Determine the way critical resources are protected if a domain loses operation quorum or if the configuration manager, Group Services, or Topology Services daemons die or hang. It is done by setting the CritRsrcProtMethod attribute of the IBM.PeerNode class (or an individual IBM.PeerNode instance).
- Specify that the peer domain must always have operational quorum. Forcing operational quorum in this way, as opposed to having the configuration resource manager calculate whether the peer domain has operation quorum, is not recommended since it means that critical resource is not protected.
- Set the active tiebreaker that the configuration resource manager uses to resolve tie situations when two or more subdomains that contain exactly half the defined nodes are competing for operational quorum. In addition, you can modify a tiebreaker definition, define a new tiebreaker, explicitly resolve a tie when the active tiebreaker type is *Operator*.

For complete syntax information on the generic RMC commands (such as **lsrsrc** and **chrsrc**), see their man pages in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Setting the critical resource protection method for a peer domain or a node in a peer domain

You can set the critical resource protection method for a peer domain by setting the **CritRsrcProtMethod** persistent attribute of the **IBM.PeerNode** resource class.

When an RSCT peer domain is partitioned into two or more subdomains, the configuration resource manager determines which subdomain has operational quorum and survives, and which others must be dissolved. If the subdomain is to be dissolved, the configuration resource manager sets the **OpQuorumState** dynamic attribute of the **PeerDomain** resource to 2 (NoQuorum).

If critical resources are active on a node that lost quorum (as indicated by the **CritRsrcActive** dynamic attribute of the **IBM.PeerNode** resource), the configuration resource manager uses a critical resource protection method on the node to ensure that critical resources are not corrupted as a result of the domain partitioning. This condition is essential, since certain applications require shared resource access. When a domain is partitioned, each subdomain is unaware of any other subdomain. Therefore, multiple subdomains might simultaneously access the shared resource and, in doing so, cause data corruption. The critical resource protection method of a node is also needed if the configuration manager, Group Services, or Topology Services daemons die or hang.

CritRsrcProtMethod persistent attribute

You can set the critical resource protection method for a peer domain by setting the **CritRsrcProtMethod** persistent attribute of the **IBM.PeerNode** resource class. By default, the same critical resource protection method is employed for all nodes of the peer domain (all instances of the **IBM.PeerNode** resource class). You can specify a different critical resource protection method for a particular node, however, by setting the **CritRsrcProtMethod** persistent attribute for just that instance of the **IBM.PeerNode** resource class.

Table 23 shows the possible settings for the **CritRsrcProtMethod** persistent attribute.

Table 23. CritRsrcProtMethod settings

Persistent attribute value	Description
0	The resource instance inherits the value from the resource class. It is the default value for the individual resource instances of **IBM.PeerNode**.
1	Performs a hard reset and restarts the system. It is the default value for the **IBM.PeerNode** resource class.
2	Halts the system.
3	Synchronizes, performs a hard reset, and restarts the system.
4	Synchronizes and halts the system.
5	None.
6	Exits and restarts the RSCT subsystems.
7	Disables all protections.

To view or set the critical resource protection method for a peer domain or a node in the peer domain, use the standard Resource Monitoring and Control (RMC) management commands - **lsrsrc** and **chrsrc**.

For example, to list the current value of the **CritRsrcProtMethod** persistent attribute for each node in the domain, use the **lsrsrc** command.

```
# lsrsrc -t IBM.PeerNode Name CritRsrcProtMethod
Name           CritRsrcProtMethod
"Davros"       0
"Rassilon"     0
"Morbius"      0
"Zagreus"      0
```

The preceding output shows that each node currently inherits the overall critical resource protection method of the peer domain. To list the domain-wide attributes, use the **lsrsrc** command with its **-c** flag.

```
# lsrsrc -c IBM.PeerNode
Resource Class Persistent Attributes for: IBM.PeerNode
resource 1:
        CommittedRSCTVersion       = ""
        ActiveVersionChanging      = 0
        OpQuorumOverride           = 0
        CritRsrcProtMethod         = 1
        OpQuorumTieBreaker         = "Operator"
        QuorumType                 = 0
        QuorumGroupName            = ""
        Fanout                     = 32
        OpFenceGroup               = ""
        NodeCleanupCommand         = ""
        NodeCleanupCriteria        = ""
        QuorumLessStartupTimeout   = 120
        CriticalMode               = 1
        NotifyQuorumChangedCommand = ""
        NamePolicy                 = 0
```

To override the default domain-wide critical resource protection method on a single node, you would use the **chrsrc** command. This next example uses the **-s** flag and a selecting string to identify the node.

```
chrsrc -s"Name=='Zagreus'" IBM.PeerNode CritRsrcProtMethod=3
```

To change the domain-wide critical resource protection method, you would use the **chrsrc** command with its **-c** flag.

```
chrsrc -c IBM.PeerNode CritRsrcProtMethod=3
```

CriticalMode persistent attribute

The **CriticalMode** persistent attribute of the **IBM.PeerNode** class can be configured to enable the critical resource protection for all nodes or to disable the critical resource protection even for active critical resources.

Table 24. CriticalMode attribute settings

Persistent attribute value	Description
0	Critical resource protection is disabled even if a node has active critical resources.
1	The **CriticalMode** attribute does not affect critical resource protection. It is the default value for **CriticalMode** attribute.
2	Critical resource protection is enabled for all nodes.

To query the current value of the **CriticalMode** attribute, run the following command:

```
lsrsrc -c IBM.PeerNode CriticalMode
```

To change the value of the **CriticalMode** attribute, run the following command:

```
chrsrc -c IBM.PeerNode CriticalMode=new_value
```

Related tasks:

"Explicitly resolving a tie when the active tiebreaker type is Operator" on page 102
When the active tiebreaker is the predefined tiebreaker Operator or a tiebreaker whose persistent attribute Type is Operator, then the configuration resource manager does not automatically resolve tie situations.

"Setting the active tiebreaker" on page 89
The **OpQuorumTieBreaker** persistent class attribute of the **IBM.PeerNode** class indicates the active tiebreaker for the peer domain.

Related information:

lsrsrc command

chrsrc command

Overriding the configuration resource manager's operational quorum calculation to force operational quorum

Exercise caution before overriding the configuration resource manager's operational quorum calculation, since it means that critical resources will not be protected by the critical resource protection method.

When a peer domain is partitioned, the configuration manager will, by default, determine which subdomain has operational quorum using the following calculation:

```
If (( 2*numNodesOnline ) > numNodesDefined )
        OpQuorumState = HasQuorum
If (( 2*numNodesOnline ) == numNodesDefined )
        OpQuorumState = PendingQuorum
        (until tiebreaker is won or lost).
If (( 2*numNodesOnline) < numNodesDefined )
        OpQuorumState = NoQuorum
```

By setting the OpQuorumOverride persistent class attribute of the IBM.PeerNode resource class, however, you can override this calculation and instead specify that the domain should always have operational quorum. If you do this, the PeerDomain resource's OpQuorumState dynamic attribute will always have the value 0 (HasQuorum). Exercise caution before overriding the configuration resource manager's operational quorum calculation, since it means that critical resources will not be protected by the critical resource protection method.

Table 25 shows the possible settings for the OpQuorumOverride persistent class attribute of the IBM.PeerNode resource class.

Table 25. OpQuorumOverride settings

Persistent class attribute value	Description
0	Determine operation quorum
1	Force operational quorum

To view or set the OpQuorumOverride persistent class attribute of the IBM.PeerNode resource class, use the standard RMC management commands **lsrsrc** and **chrsrc**.

For example, to list the current value of the CritRsrcProtMethod persistent attribute, you would use the **lsrsrc** command with its **-c** flag:

```
# lsrsrc -c IBM.PeerNode
Resource Class Persistent Attributes for: IBM.PeerNode
resource 1:
        CommittedRSCTVersion  = ""
        ActiveVersionChanging = 0
        OpQuorumOverride      = 0
        CritRsrcProtMethod    = 1
        OpQuorumTieBreaker    = "Fail"
```

To force operational quorum for the peer domain, you would use the **chrsrc** command with its **-c** flag.

```
chrsrc -c IBM.PeerNode OpQuorumOverride=1
```

For complete syntax information on the **lsrsrc** and **chrsrc** commands, see their man pages in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Determining how the configuration resource manager will resolve tie situations when calculating operational quorum

In the case of a tie in which the peer domain has been partitioned into two subdomains containing exactly half of the defined nodes, the configuration resource manager uses a tie-breaker resource (an instance of the IBM.TieBreaker resource class) to determine which subdomain has operational quorum.

When a peer domain is partitioned, the configuration resource manager must determine which subdomain has operational quorum and so will survive, and which subdomain will be dissolved. Often, this is a case of determining which of the subdomains has more than half of the nodes. In the case of a tie in which the peer domain has been partitioned into two subdomains containing exactly half of the defined nodes, the configuration resource manager uses a tie-breaker resource (an instance of the IBM.TieBreaker resource class) to determine which subdomain has operational quorum. A tie situation also occurs when exactly half the nodes of a domain are online and the other half are inaccessible. You can have a number of IBM.TieBreaker resources defined, but only one can be active at any one time.

Setting the active tiebreaker:

The **OpQuorumTieBreaker** persistent class attribute of the **IBM.PeerNode** class indicates the active tiebreaker for the peer domain.

A peer domain can have a number of tiebreakers (**IBM.TieBreaker** resources) defined, but only one can be active at a time. If there are multiple subdomains with the same number of nodes, the configuration resource manager uses this active tiebreaker after domain partitioning to determine which subdomain has operational quorum. Apart from the predefined tiebreaker resources, you can also define your own. Table 26 describes the predefined tiebreakers: operator, fail, and success.

*Table 26. Predefined tiebreakers (**IBM.TieBreaker** resources)*

Tiebreaker	Description
Operator	The system administrator resolves the tie by starting the **ResolveOpQuorumTie** action of the **IBM.PeerDomain** resource class. Until the administrator explicitly breaks the tie, neither domain has operational quorum. The **OpQuorumState** dynamic attribute of the **PeerDomain** resource is 1 (PendingQuorum) until the administer starts the **ResolveOpQuorumTie** action.
Fail	A pseudo tiebreaker in that it does not resolve the tie situation. Neither subdomain has operational quorum. The **OpQuorumState** dynamic attribute of each **PeerDomain** resource is 2 (NoQuorum). If critical resources are active on a domain that lost quorum (as indicated by the **CritRsrcActive** dynamic attribute of the **PeerDomain** resource), the configuration resource manager uses a critical resource protection method on the node to ensure that critical resources are not corrupted as a result of the domain partitioning.
Success	A pseudo tiebreaker that is always successful in resolving a tie so that each of the subdomain has operational quorum. The **OpQuorumState** dynamic attribute of each **PeerDomain** resource is 0 (HasQuorum). Therefore, if critical resources are active on a domain, they are not protected.

To view or set the active tiebreaker (**OpQuorumTieBreaker** persistent class attribute of the **IBM.PeerNode** class), use the standard RMC management commands: **lsrsrc** and **chrsrc**.

For example, to list the current active tiebreaker, use the **lsrsrc** command with its **-c** flag.

```
# lsrsrc -c IBM.PeerNode OpQuorumTieBreaker
Resource Class Persistent and Dynamic Attributes for: IBM.PeerNode
resource 1:
        OpQuorumTieBreaker = "Fail"
```

The preceding output shows that the current active tiebreaker is *Fail*. To list the names of all of the available tiebreaker resources, specify *Name* as a parameter on the **lsrsrc** command.

```
# lsrsrc IBM.TieBreaker Name
Resource Persistent and Dynamic Attributes for: IBM.TieBreaker
resource 1:
        Name = "Operator"
```

```
resource 2:
        Name = "Fail"
resource 3:
        Name = "Success"
```

To make the operator tiebreaker the active tiebreaker, use the **chrsrc** command with its **-c** flag.

```
CT_MANAGEMENT_SCOPE=2 chrsrc -c IBM.PeerNode OpQuorumTieBreaker="Operator"
```

If you set the active tiebreaker to *Operator*, and a tie situation occur, you need to manually resolve the tie by starting the **ResolveOpQuorumTie** action of the **IBM.PeerDomain** resource class.

For complete syntax information about the **lsrsrc** and **chrsrc** commands, see their man pages in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Related tasks:

"Setting the critical resource protection method for a peer domain or a node in a peer domain" on page 86

You can set the critical resource protection method for a peer domain by setting the **CritRsrcProtMethod** persistent attribute of the **IBM.PeerNode** resource class.

Modifying a tiebreaker definition:

A tiebreaker (**IBM.TieBreaker** resource) has a number of persistent resource attributes that you can set to configure the behavior of the tiebreaker.

To view or set these persistent class attributes, use the standard RMC management commands **lsrsrc** and **chrsrc**.

For example, to list the current persistent attribute values for all defined tiebreakers, use the **lsrsrc** command.

```
# lsrsrc IBM.TieBreaker
Resource Persistent Attributes for: IBM.TieBreaker
resource 1:
        Name                = "Operator"
        Type                = "Operator"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
resource 2:
        Name                = "Fail"
        Type                = "Fail"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
resource 3:
        Name                = "Success"
        Type                = "Success"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
```

To limit the output of the **lsrsrc** command to display the persistent attribute values for only a particular tiebreaker resource, use the **-s** flag and a selection string that identifies the particular tiebreaker resource.

```
# lsrsrc -s"Name=='Operator'" IBM.TieBreaker
Resource Persistent Attributes for: IBM.TieBreaker
resource 1:
        Name                = "Operator"
        Type                = "Operator"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
```

To obtain more information about any of these persistent attributes, use the **lsrsrcdef** command. To change the persistent attributes of a tiebreaker, the tiebreaker must not be the active tiebreaker. The **OpQuorumTieBreaker** persistent class attribute of the **IBM.PeerNode** class identifies the active tiebreaker. If you are not sure whether the tiebreaker that you want to modify is the active tiebreaker or not, check the value of the **OpQuorumTieBreaker** persistent class attribute.

```
# lsrsrc -c IBM.PeerNode OpQuorumTieBreaker
Resource Class Persistent and Dynamic Attributes for: IBM.PeerNode
resource 1:
        OpQuorumTieBreaker = "Fail"
```

Until the tiebreaker is not the active tiebreaker, you can modify its persistent resource attributes by using the **chrsrc** command. To identify a particular tiebreaker, use the **chrsrc** command with the **-s** flag followed by a selection string that identifies the tiebreaker resource. For example:

```
chrsrc -s"Name=='Operator'" IBM.TieBreaker ReleaseRetryPeriod=30
```

For complete syntax information about the **lsrsrc** and **chrsrc** commands, see their man pages in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides

Defining a new tiebreaker:

In addition to the predefined tiebreakers, you can also create your own tiebreaker. To do so, use the **mkrsrc** command to define a new **IBM.TieBreaker** resource.

Attention: When defining tiebreaker resources, be aware that the disk on which **IBM.TieBreaker** resources are stored must not also be used to store file systems.

Before you define a tiebreaker, determine which persistent attributes are required when defining an **IBM.TieBreaker** resource. This information can be returned by issuing the **mkrsrc -e** command. The **-e** flag causes the **mkrsrc** command to display two examples of suitable command-line input when defining a resource. One example shows the suitable command-line input for required attributes only. The other example shows the suitable command-line input for both required and optional attributes. For example:

```
# mkrsrc -e IBM.TieBreaker
Sample mkrsrc command with required attributes:
mkrsrc IBM.TieBreaker Type=char_ptr Name=char_ptr

Sample mkrsrc command with required and optional attributes:
mkrsrc IBM.TieBreaker Type=char_ptr Name=char_ptr ReprobeData=char_ptr PreReserv
eWaitTime=uint32 DeviceInfo=char_ptr NodeInfo=sd_ptr_array PostReserveWaitTime=u
int32 HeartbeatPeriod=uint32 ReleaseRetryPeriod=uint32
```

To obtain more information about any of the attributes of an **IBM.TieBreaker** resource, use the **lsrsrcdef** command. The two attributes that are required for defining an **IBM.TieBreaker** resource are: the **Name** attribute and the **Type** attribute.

- The **Type** attribute is the name of one of the available tiebreaker types. The available tiebreaker types depend on your operating system and machine architecture. Possible types are:

  DISK Enables you to specify a small computer system interface (SCSI) or SCSI-like physical disk by using an AIX device name, assuming that the SCSI disk is shared by one or more nodes of the peer domain. tiebreaker reservation is done by the SCSI reserve or persistent reserve command. If you are creating a tiebreaker of this type, set the **DeviceInfo** persistent resource attribute to identify the physical disk. Only SCSI and SCSI-like physical disks are supported. Physical disks attached through Fibre Channel and iSCSI are suitable.

  This tiebreaker type is specific to AIX.

  ECKD™
  Assumes that an ECKD DASD is shared by all nodes of the cluster. tiebreaker reservation is done by using the ECKD reserve command. If you are creating a tiebreaker of this type, set the **DeviceInfo** persistent resource attribute to indicate the ECKD device number.

  This tiebreaker type is specific to Linux on the System z hardware platform.

  EXEC Indicates that a script provided by an RSCT exploiter product resolves the tie situation.

  Fail Is a pseudo tiebreaker type that always fails to reserve the tiebreaker.

  Operator
  Requests a decision from the system operator or administrator. The operator runs a decision by starting the **ResolveOpQuorumTie** action.

  Success
  Is a pseudo tiebreaker type that is always successful in reserving the tiebreaker.

  SCSI Assumes that an SCSI-disk is shared by two or more nodes of the peer domain. Tiebreaker reservation is done by using the SCSI reserve command. If you are creating a tiebreaker of this type, set the **DeviceInfo** persistent resource attribute to identify the SCSI device.

  This tiebreaker type is specific to Linux on the Power Systems and System x hardware platforms.

  SCSIPR
  Uses SCSI-3 persistent reservations on an SCSI disk storage device as a tie breaking mechanism. Sometimes, the peer domain is partitioned into two subdomains that contain exactly half of the defined nodes and a tie occurs. In such cases, the subdomain that gets an exclusive persistent reservation of the SCSI disk storage device obtains the operational quorum.

  This tiebreaker type is introduced by Tivoli System Automation for Multiplatforms (TSAMP) 3.2.1.2 and works for Linux on the System x hardware platform and for AIX on the Power Systems platform. It is supported on Red Hat Enterprise Linux (RHEL) 5, RHEL 6, SUSE Linux Enterprise Server (SLES) 10, SLES 11, and the AIX 6.1 platform.

  The tiebreaker types that are available for your operating system and machine architecture are listed in the **AvailableTypes** class attribute of the **IBM.TieBreaker** resource class. To list the available tiebreaker types, use the **lsrsrc** command with its **-c** flag.

```
# lsrsrc -c IBM.TieBreaker AvailableTypes
Resource Class Persistent and Dynamic Attributes for: IBM.TieBreaker
resource 1:
        AvailableTypes = {["Operator",""],["Fail",""],["Success",""]}
```

  If the **lsrsrc** command as shown in the preceding example is issued on a Linux zSeries machine, the output shows ECKD as one of the available types. If issued on a Linux xSeries machine, the output shows SCSI as an available type. If issued on an AIX machine, the output shows DISK as an available type.

- The **Name** attribute is a null-terminated string that you use to identify this tiebreaker. It is the value you use when setting the **OpQuorumTieBreaker** persistent class attribute of the **IBM.PeerNode** resource class to activate the tiebreaker.

After you understand the values you want to assign to the persistent attributes that are required for definition (and any attributes that are optional for definition that you want to specify), you define the **IBM.TieBreaker** resource by using the **mkrsrc** command. For example:

```
mkrsrc IBM.TieBreaker Name=OpQuorumTieBreaker Type=Operator
```

For complete syntax information about the **lsrsrcdef**, **lsrsrc** and **mkrsrc** commands, see their man pages in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Table 27 shows the shared disk environments that are supported for tiebreaker on AIX systems.

Table 27. Supported shared disk environments for tiebreaker on AIX

Disk families	Disk attachment	Pathing	Reservation type	Supported?	Notes
DS3000, DS4000®, DS5000, DS5020, DS6000™, DS8000®					1, 2
	Directly attached to client	DS3K (MPIO). DS4K, DS5K, DS6K, DS8K (MPIO, RDAC, SDDPCM). DS5020 (MPIO and SDDPCM).	SCSI-2	yes	3, 6
	Directly attached to client	DS3K (MPIO). DS4K, DS5K, DS6K, DS8K (MPIO, RDAC, SDDPCM). DS5020 (MPIO and SDDPCM).	SCSI-3	yes	3, 6, 9
	VIOS (vSCSI, default)	All DS families (MPIO)	SCSI-2	no	4
	VIOS (vSCSI, client reserve)	All DS families (MPIO and SDDPCM)	SCSI-2	yes	7, 8
	VIOS (vSCSI, default)	All DS families (MPIO)	SCSI-3	no	5
	VIOS (NPIV)	DS3K (MPIO). DS4K, DS5K, DS6K, DS8K (MPIO, RDAC, SDDPCM). DS5020 (MPIO and SDDPCM).	SCSI-2	yes	3, 6, 7
	VIOS (NPIV)	DS3K (MPIO). DS4K, DS5K, DS6K, DS8K (MPIO, RDAC, SDDPCM). DS5020 (MPIO and SDDPCM).	SCSI-3	yes	3, 6, 7, 9

Notes:

1. Only DS3400 is supported as a storage solution for VIOS attachment from the DS3000 family.
2. DS5020 is not a supported storage solution for VIOS attachment.
3. MPIO (**devices.common.IBM.mpio.rte**) is the default AIX path control module and must be used instead of the FCPARRAY (also known as RDAC, **devices.fcp.disk.array.rte**)
4. In this default environment, the reservation is emulated by the VIOS and does not pass the SCSI commands to the physical device.
5. VIOS does not support SCSI-3 (also known as persistent reservation) in a VSCSI environment.
6. FCPARRAY is not supported on the DS5000/DS5020 families.
7. SDDPCM is not supported on VIOS for the DS4000 and DS5000/5020 families.
8. Client reserve refers to the configuration where the VIOS does not pass SCSI commands to the physical device, but rather allows the virtual client to do so.
9. Limited to a two-node peer domain. Physical volumes need to have attributes modified, so each node has a different **PR_key_value** and each has **reserve_policy=PR_exclusive**.

Table 28 on page 94 shows the shared disk environments that are supported for tiebreaker on Linux systems.

Table 28. Supported shared disk environments for tiebreaker on Linux

Disk families	Disk attachment	Pathing	Reservation type	Supported?	Notes
DS3000, DS4000, DS5000, DS5020, DS6000, DS8000					1, 2
	Directly attached to client	DS3K, DS4K, DS5K, DS5020 (RDAC)	SCSI-2	yes	3
	Directly attached to client	DS3K, DS4K, DS5K, DS5020 (RDAC and DM-MP). DS6K (DM-MP, SUSE only). DS8K (DM-MP and SDD).	SCSI-3	yes	4
	VIOS (vSCSI, default)	All DS families (MPIO)	SCSI-2	no	5
	VIOS (vSCSI, client reserve)	All DS families (MPIO)	SCSI-2	no	6
	VIOS (vSCSI, default)	All DS families (MPIO)	SCSI-3	no	7
	VIOS (NPIV)	DS3K, DS4K, DS5K, DS5020 (RDAC)	SCSI-2	yes	8
	VIOS (NPIV)	DS3K, DS4K, DS5K, DS5020 (RDAC and DM-MP). DS6K (DM-MP, SuSE only). DS8K (DM-MP and SDD).	SCSI-3	no	7

Notes:
1. Only DS3400 is supported as a storage solution for VIOS attachment from the DS3000 family.
2. DS5020 is not a supported storage solution for VIOS attachment.
3. Pathing drivers supported on the Power Systems and System x hardware platforms.
4. SCSI-3 reserve is supported through the SCSIPR tiebreaker module for RSCT in AIX and Linux environments, provided the disk subsystems are correctly configured for SCSI-3.
5. In this environment, the reservation is emulated by the VIOS. If another system is directly attached to the disks, the reservation can be taken by that system.
6. VIOS does not support SCSI-2 reserves for Linux operating system on IBM Power® virtual clients.
7. VIOS does not support SCSI-3 (also known as persistent reservation) in a VSCSI environment.
8. Linux on the Power virtual client is the virtual client operating system.

Related tasks:

"Displaying attribute definition information for a resource or a resource class" on page 135
You can display attribute definition information for a resource or a resource class by issuing the **lsrsrcdef** command with the name of a resource class.

"Creating an ECKD tiebreaker" on page 96
The ECKD tiebreaker type is specific to Linux on zSeries. To create an ECKD tiebreaker object, set the DeviceInfo persistent resource attribute to indicate the ECKD device number.

"Creating an SCSI tiebreaker" on page 97
The SCSI tiebreaker type is specific to Linux on the System x and Power Systems hardware platforms. If you want to create an SCSI tiebreaker object, specify the SCSI device by using the **DeviceInfo** persistent resource attribute. If the SCSI configuration is different between nodes, you can also use the **NodeInfo** persistent resource attribute to reflect those differences.

Creating a DISK tiebreaker:

The DISK tiebreaker type is specific to AIX. To create a DISK tiebreaker object, set the **DeviceInfo** persistent resource attribute to indicate the AIX device name or the PVID of the disk. The AIX device name must specify a SCSI or SCSI-like physical disk that is shared by all nodes of the peer domain. Disks with the DISK tie-breaker type only support these **mpio** pathing modules: FCPARRAY (for AIX 5.3), MPIO, and SDDPCM (with the single-path reserve policy). Any non-**mpio** disks can be supported if SCSI-2 reserve/release is supported.

Physical disks attached through Fiber Channel and iSCSI can serve as a DISK tiebreaker. However, IDE hard disks do not support the SCSI protocol and cannot serve as a DISK tiebreaker. Logical volumes also cannot serve as a DISK tiebreaker.

This type of tiebreaker can use a reserve/release mechanism or a persistent reserve. For reserve/release and persistent reserve, the disk needs to be re-reserved periodically to hold the reservation. For this reason, IBM recommends that you also specify the **HeartbeatPeriod** persistent resource attribute when creating a tiebreaker of this type. The **HeartbeatPeriod** persistent resource attribute defines the interval at which the reservation is retried.

Attention: When defining tiebreaker resources, be aware that the disk on which **IBM.Tiebreaker** resources are stored should not also be used to store file systems.

To print every known physical volume in the system along with its physical disk name, enter the **lspv** command:

```
lspv
```

Output similar to the following is displayed:

```
hdisk0          000000371e5766b8                    rootvg          active
hdisk1          000069683404ed54                    None
```

To verify that a disk is a SCSI or SCSI-like disk and so a suitable candidate for a DISK tiebreaker, use the **lsdev** command. For example:

```
lsdev -C -l hdisk0
```

Output similar to the following is displayed

```
hdisk0 Available 10-60-00-0,0 16 Bit SCSI Disk Drive
```

In order to serve as a tie-breaker disk using reserve/release, the disk must be shared by all nodes of the peer domain. Check the physical volume ID returned by the **lspv** command to determine if the disk is shared between nodes (in the preceding output for the **lspv** command, the physical volume ID is listed in the second column; the volume ID for **hdisk0** is **000000371e5766b8**.) Be aware, however, that AIX remembers all disks that have been attached to the system, and the disks listed by the **lspv** command may no longer be attached. If such a disk was moved to another machine, it might appear that the disk is shared, when in fact it is no longer attached to the original machine.

The disk on which IBM.Tiebreaker resources are stored should not also be used to store file systems. If the nodes of the cluster share more than one disk, it may be difficult to determine which one is the tie-breaker disk, and which one is used for regular data. The output from the **lsdev** command shows the SCSI address associated with the disk. (In the preceding output for the **lsdev** command, the SCSI address is listed in the third column; the SCSI address for **hdisk0** is **10-60-00-0,0**.) This information will help you identify the correct disk if you are aware of the disk's address prior to its installation.

If the tie-breaker disk will be using persistent reserves, an additional step needs to be performed on each node.

1. Run the **chdev** command on the first node:

   ```
   chdev -l hdisk -a PR_key_value=0x0001 -a reserve_policy=PR_exclusive
   ```

2. The **PR_key_value** must be a different value on the second node. Run the **chdev** command on the second node:

   ```
   chdev -l hdisk -a PR_key_value=0x0002 -a reserve_policy=PR_exclusive
   ```

Once you know the device name, you can issue the **mkrsrc** command, as follows:

```
mkrsrc IBM.TieBreaker Name=disktb Type=DISK DeviceInfo="DEVICE=/dev/hdisk0" \
       HeartbeatPeriod=30
```

You can also use the PVID (instead of the device name) to specify the tiebreaker, as follows:

```
mkrsrc IBM.TieBreaker Name=disktb Type=DISK DeviceInfo="PVID=000000371e5766b8" \
       HeartbeatPeriod=30
```

Creating an ECKD tiebreaker:

The ECKD tiebreaker type is specific to Linux on zSeries. To create an ECKD tiebreaker object, set the DeviceInfo persistent resource attribute to indicate the ECKD device number.

This type of tiebreaker uses a reserve/release mechanism and needs to be re-reserved periodically to hold the reservation. For this reason, we strongly recommend that you also specify the HeartbeatPeriod persistent resource attribute when creating a tiebreaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation is retried.

Attention: When defining tiebreaker resources, be aware that the disk on which IBM.Tiebreaker resources are stored should not also be used to store file systems.

If you are using the SUSE Linux Enterprise Server 9 (SLES 9), you can obtain the ECKD device number by entering the command:

```
/sbin/lsdasd
```

Output similar to the following is displayed. In the following example output, **bold** text is used to highlight the ECKD device number.

```
0.0.0100(ECKD) at ( 94: 0) is dasda : active at blocksize: 4096, 601020 blocks, 2347 MB
0.0.0101(FBA ) at ( 94: 4) is dasdb : active at blocksize: 512, 2454165 blocks, 1198 MB
```

For other Linux distributions, you can obtain the device number by entering the command:

```
cat /proc/dasd/devices
```

Output similar to the following is displayed. In the following example output, **bold** text is used to highlight the ECKD device number.

```
50dc(ECKD) at ( 94:  0) is    : active at blocksize: 4096, 601020 blocks, 2347 MB
50dd(ECKD) at ( 94:  4) is    : active at blocksize: 4096, 601020 blocks, 2347 MB
50de(ECKD) at ( 94:  8) is    : active at blocksize: 4096, 601020 blocks, 2347 MB
50df(ECKD) at ( 94: 12) is    : active at blocksize: 4096, 601020 blocks, 2347 MB
```

Once you know the device number, you can issue the **mkrsrc** command.

```
mkrsrc IBM.TieBreaker Name=eckdtest Type=ECKD DeviceInfo="ID=50dc" \
       HeartbeatPeriod=30
```

Related tasks:

"Defining a new tiebreaker" on page 91
In addition to the predefined tiebreakers, you can also create your own tiebreaker. To do so, use the **mkrsrc** command to define a new **IBM.TieBreaker** resource.

Creating an EXEC tiebreaker:

You can only create a tiebreaker of this type if an RSCT exploiter product provides a script or executable designed to resolve a tie situation. If an exploiter product has provided such a script or executable, and it has been installed on all nodes of your cluster, you can create an EXEC tiebreaker object using the **mkrsrc** command.

For the EXEC tiebreaker type, the **DeviceInfo** attribute should specify the path to the script or executable and any program arguments. For example, to create the tiebreaker named **MyTB** when the provided executable is **/usr/sbin/rsct/bin/tiebreaker**, you would enter:

```
mkrsrc IBM.TieBreaker Type="EXEC" Name="MyTB" \
       DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/tiebreaker myArg=123'
```

Creating an SCSI tiebreaker:

The SCSI tiebreaker type is specific to Linux on the System x and Power Systems hardware platforms. If you want to create an SCSI tiebreaker object, specify the SCSI device by using the **DeviceInfo** persistent resource attribute. If the SCSI configuration is different between nodes, you can also use the **NodeInfo** persistent resource attribute to reflect those differences.

Disks for the SCSI tiebreaker type can support three device types: **RDAC.WWN**, **WWID**, and the SCSI identifiers that use the **CHAN**, **HOST**, **ID**, and **LUN** keywords. Use the **RDAC.WWN** keyword when the disks are being used with the LSI RDAC **mpio** pathing module. For other pathing modules that support SCSI-2 reservations, use the **WWID** keyword. Any non-**mpio** disks can be used if SCSI-2 reserve/release is supported by using the **CHAN**, **HOST**, **ID**, or **LUN** keyword.

Note: Any disk that supports SCSI-2 release/reserve can be used by specifying the **CHAN**, **HOST**, **ID**, or **LUN** keyword, even if the LSI RDAC pathing module is in use or if another pathing module is being used. In other words, if you can specify **RDAC.WWN** or **WWID** successfully, you can also use the **CHAN**, **HOST**, **ID**, and **LUN** keywords instead.

The SCSI tiebreaker type uses a reserve and release mechanism and needs to be reserved again periodically to hold the reservation. For this reason, IBM recommends that you also specify the **HeartbeatPeriod** persistent resource attribute when creating a tie breaker of this type. The **HeartbeatPeriod** persistent resource attribute defines the interval at which the reservation is tried again.

Attention: When defining tie breaker resources, be aware that the disk on which **IBM.TieBreaker** resources are stored must not also be used to store file systems.

To obtain the identifiers for an SCSI device, enter:

```
sginfo -l
```

The following output is displayed:

```
/dev/sda /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf /dev/sdg /dev/sdh
/dev/sdi /dev/sdj /dev/sdk /dev/sdl /dev/sdm /dev/sdn /dev/sdo /dev/sdp
/dev/sdq /dev/sdr /dev/sds /dev/sdt /dev/sdu /dev/sdv /dev/sdw /dev/sdx
/dev/sdy /dev/sdz /dev/sdaa /dev/sdab /dev/sdac /dev/sdad
/dev/sg0 [=/dev/sda  scsi2 ch=0 id=1 lun=0]
/dev/sg1 [scsi2 ch=0 id=8 lun=0]
/dev/sg2 [=/dev/sdb  scsi4 ch=0 id=0 lun=0]
/dev/sg3 [=/dev/sdc  scsi4 ch=0 id=0 lun=1]
/dev/sg4 [=/dev/sdd  scsi4 ch=0 id=0 lun=2]
/dev/sg5 [=/dev/sde  scsi4 ch=0 id=0 lun=3]
/dev/sg6 [=/dev/sdf  scsi4 ch=0 id=0 lun=4]
/dev/sg7 [=/dev/sdg  scsi4 ch=0 id=0 lun=5]
/dev/sg8 [=/dev/sdh  scsi4 ch=0 id=0 lun=6]
/dev/sg9 [=/dev/sdi  scsi4 ch=0 id=0 lun=7]
```

Once you know the identifiers for the SCSI device, you can issue the **mkrsrc** command. If the SCSI configuration is the same on all nodes, identify the SCSI device using the DeviceInfo persistent resource attribute. For example, if you use **/dev/sdc(/dev/sg3)** as the SCSI device:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI DeviceInfo="ID=0 LUN=1 HOST=4,CHAN=0"  \
 HeartbeatPeriod=30
```

Because the SCSI configuration can be different between nodes (even if the target device is the same), you might need to reflect differences between nodes by using the NodeInfo persistent resource attribute. For example, suppose that an SCSI device is connected to two nodes and has the following SCSI identifiers:

```
node1: HOST=0 CHAN=0 ID=4 LUN=0
node2:  HOST=1 CHAN=2 ID=4 LUN=0
```

You would create the tie breaker object by entering the following **mkrsrc** command:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI DeviceInfo="ID=4 LUN=0" \
       NodeInfo='{["node1", "HOST=0,CHAN=0"], ["node2", "HOST=1 CHAN=2"]}' \
       HeartbeatPeriod=30
```

For each node, the configuration resource manager merges the DeviceInfo string with the NodeInfo string. In the preceding example, the merged string for **node1** will be **"ID=4 LUN=0 HOST=0 CHAN=0"**. Any duplicate keywords specified in the **DeviceInfo** and **NodeInfo** strings are allowed, and the last one will be used. So the preceding command could also have been specified as:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI DeviceInfo="ID=4 LUN=0 HOST=0,CHAN=0" \
NodeInfo='["node2", "HOST=1 CHAN=2"]}' HeartbeatPeriod=30
```

This simplification can be useful when the SCSI identifiers are the same for many nodes. You have to use the **NodeInfo** attribute only to specify the nodes that are different.

If the RDAC driver is installed, an SCSI tie breaker can also be specified by using the worldwide number (WWN). To obtain the WWN for each disk, enter one of the following commands:

```
cat /proc/scsi/mpp/* | grep "^[ ]*Lun"

cat /proc/scsi/mpp/* | awk '/^[ ]*Lun/{print}'
```

The following output is displayed:

```
Lun Information
    Lun #0 - WWN:600a0b80000cf57b00000007406ec4c6
       LunObject: h4c0t0l0                              CurrentOwningPath: A
    Lun #1 - WWN:600a0b80000f01380000000d406ec4c5
       LunObject: h4c0t0l1                              CurrentOwningPath: B
    Lun #2 - WWN:600a0b80000cf57b00000009406ec512
       LunObject: h4c0t0l2                              CurrentOwningPath: A
    Lun #3 - WWN:600a0b80000f01380000000f406ec4f9
       LunObject: h4c0t0l3                              CurrentOwningPath: B
    Lun #4 - WWN:600a0b80000cf57b0000000b406ec542
       LunObject: h4c0t0l4                              CurrentOwningPath: A
    Lun #5 - WWN:600a0b80000f013800000011406ec529
       LunObject: h4c0t0l5                              CurrentOwningPath: B
    Lun #6 - WWN:600a0b80000cf57b0000000d406ec56e
       LunObject: h4c0t0l6                              CurrentOwningPath: A
    Lun #7 - WWN:600a0b80000f013800000013406ec553
       LunObject: h4c0t0l7                              CurrentOwningPath: B
    Lun #8 - WWN:600a0b80000cf57b0000000f406ec598
       LunObject: h4c0t0l8                              CurrentOwningPath: A
```

In this sample output, each disk is associated with a WWN value and, on the following line, a **LunObject** value. For example, the following part of the output shows the WWN and **LunObject** values associated with Lun #6:

```
    Lun #6 - WWN:600a0b80000cf57b0000000d406ec56e
       LunObject: h4c0t0l6                              CurrentOwningPath: A
```

The format of the **LunObject** value (**h4c0t0l6** for **Lun #6** above, for example) is:
h*host_number***c***channel_number***t***scsi_id***l***lun*

To create a tie breaker object by using the **RDAC.WWN** keyword, issue the **mkrsrc** command. For example:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI \
DeviceInfo="RDAC.WWN=600a0b80000cf57b0000000d406ec56e" \
HeartbeatPeriod=30
```

To create a tie breaker object by using a WWID, issue the **mkrsrc** command. For example:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI DeviceInfo="WWID=600a0b80000cf57b0000000d406ec56e" \
HeartbeatPeriod=30
```

It is assumed that this unique identifier of the disk (the WWID) is already known.

Related tasks:

"Defining a new tiebreaker" on page 91
In addition to the predefined tiebreakers, you can also create your own tiebreaker. To do so, use the **mkrsrc** command to define a new **IBM.TieBreaker** resource.

Creating an SCSIPR tiebreaker:

The small computer system interface persistent reservation (SCSIPR) tiebreaker type is supported by Linux on System x hardware platform and AIX on the Power Systems platform. It is introduced by IBM Tivoli System Automation for MultiPlatforms 3.2.1.2.

The SCSI disk storage device to be used by the SCSIPR tiebreaker must support the SCSI-3 persistent reserve protocol with the Write Exclusive Registrants Only reservation type. This device must be shared by all the systems in the peer domain, and all systems must be able to reserve the device by using the persistent reserve protocol.

The SCISPR tiebreaker uses the **sg_persist** utility. Use the following commands to check whether it is already installed on all of the systems of the peer domain:

```
which sg_persist
rpm -qf /usr/bin/sg_persist
```

In Linux environment, if the **sg_persist** utility is not installed yet, you must install the appropriate Linux package.

- Red Hat Enterprise Linux (RHEL) 5, RHEL 6, SUSE Linux Enterprise Server (SLES) 11: sg3_utils*.rpm
- SLES 10: scsi*.rpm

When creating a tiebreaker of the SCSIPR type, use the **DeviceInfo** persistent resource attribute to specify the SCSI disk storage device to be used by the tiebreaker. If the SCSI configuration is different between the peer domain systems, use the **NodeInfo** persistent resource attribute to determine those differences.

The SCSIPR tiebreaker uses the reserve and release mechanism, and therefore, needs to be reserved periodically. To hold this reservation, specify the **HeartbeatPeriod** persistent resource attribute when creating a tiebreaker of this type. The **HeartbeatPeriod** persistent resource attribute defines the interval at which the reservation is tried again.

Attention: When defining the tiebreaker resources, the disk on which **IBM.Tiebreaker** resources are stored must not be used to store the file systems.

Use one of the following methods to identify the SCSI disk storage device to be used by the tiebreaker in the **DeviceInfo** persistent resource attribute:

- DEVICE=*generic_or_disk_device_name*

  For example:

  ```
  # mkrsrc IBM.TieBreaker Name="mySCSIPRTieBreaker" Type=SCSIPR
  DeviceInfo="DEVICE=/dev/sd20" HeartbeatPeriod=5
  ```

- HOST=*h* CHAN=*c* ID=*i* LUN=*l*

  For example:

  ```
  # mkrsrc IBM.TieBreaker Name="mySCSIPRTieBreaker" Type=SCSIPR
  DeviceInfo="HOST=1 CHAN=0 ID=8 LUN=0" HeartbeatPeriod=5
  # lsrsrc IBM.TieBreaker
  ```

  The following output is displayed:

```
Resource Persistent Attributes for IBM.TieBreaker
resource 1:
        Name                = "mySCSIPRTieBreaker"
        Type                = "SCSIPR"
        DeviceInfo          = "HOST=1 CHAN=0 ID=8 LUN=0"
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 5
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
        ActivePeerDomain    = "subhra"
```

- WWID=*wwid_as_displayed_by_the_system*

  For example:

```
# mkrsrc IBM.TieBreaker Name="mySCSIPRTieBreaker" Type=SCSIPR
DeviceInfo="WWID=36005076303ffc4d20000000000001153"
HeartbeatPeriod=5
# lsrsrc IBM.TieBreaker
```

  The following output is displayed:

```
resource 1:
        Name                = "Fail"
        Type                = "Fail"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
        ActivePeerDomain    = "RPD"
resource 2:
        Name                = "mySCSIPRTieBreaker"
        Type                = "SCSIPR"
        DeviceInfo          = "WWID=36005076303ffc4d20000000000001153"
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 5
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
        ActivePeerDomain    = "RPD"
resource 3:
        Name                = "Operator"
        Type                = "Operator"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
        ActivePeerDomain    = "RPD"
```

- RDAC.WWN=*wwn_as_displayed_by_the_system_when_using_LSI_RDAC_multi-path_driver*

  For example:

```
# mkrsrc IBM.TieBreaker Name="mySCSIPRTieBreaker" Type=SCSIPR
DeviceInfo="RDAC.WWN=600a0b80000cf57b00000042474304bc"
HeartbeatPeriod=5
```

Determining whether shared disks can be used as tiebreaker disks:

You can use the **tb_break** command to determine whether disks can be used for a shared disk tiebreaker. This command is available with RSCT and can be used with AIX, Linux (Power Systems servers, System

x platform, and System z platform), and Solaris. The **tb_break** command can be used only to determine whether a specific disk can accept an SCSI-2 reservation. It cannot determine whether multiple systems in the domain have access to the disk.

The **tb_break** command is in the **/usr/sbin/rsct/bin** directory. To use it, you must specify:

1. An action such as reserve, to lock the disk, or release, to unlock the disk.
2. The tiebreaker type: **DISK**, **ECKD**, or **SCSI**.
3. The device. There are different ways to specify the device, depending on the operating system:
   - **DISK**

     This tiebreaker type is for AIX. You can use the device name (**/dev/hdisk5**) or the PVID of the disk, along with the **DEVICE** keyword or the **PVID** keyword as the device.
   - **ECKD**

     This tiebreaker type is for Linux on the System z hardware platform.
   - **SCSI**

     This tiebreaker type is for Linux and Solaris. For the Power Systems servers and the System x hardware platforms, there are two ways to determine the device. You can specify the SCSI bus attributes by using the **CHAN**, **HOST**, **ID**, and **LUN** keywords, or you can specify a unique identification number. A specification that uses SCSI bus attributes might be similar to the following sample:

     ```
     "HOST=n CHAN=n ID=n LUN=n"
     ```

     You can specify a unique identification number with the **RDAC.WWN** keyword or the **WWID** keyword. Use **RDAC.WWN** when the SAN disk is being used with the LSI RDAC driver for multipathing. The **RDAC.WWN** keyword indicates **tb_break** must verify that the id is valid. Use **WWID** if the SAN disk is being used with a pathing driver which supports the SCSI-2 specification. This driver must provide a method for retrieving the unique identifier of the disk. An example is the EMC PowerPath driver, though **WWID** is not limited to EMC SANs. On Solaris, use the **DEVICE** keyword to specify the device name that identifies the disk.
   - **SCSIPR**

     This tiebreaker type is for Linux on System x hardware platform and AIX on the Power Systems platform.

     Perform the following steps on all of the remote peer systems to verify whether all the systems support the SCSIPR tiebreaker correctly with the chosen SCSI disk storage device:

     a. Reserve the disk device by using the **tb_break** command:

        ```
        /usr/sbin/rsct/bin/tb_break -v -l -t SCSIPR DeviceInfo_device_specification_for_this_system
        ```

        This command reserves the disk device.

     b. Try to reserve the same disk device by using the **tb_break** command on all of the other peer domain systems:

        ```
        /usr/sbin/rsct/bin/tb_break -v -l -t SCSIPR DeviceInfo_device_specification_for_this_system
        ```

        This command fails to reserve the disk device because it is already exclusively reserved by the first system.

     c. Release the disk device by using the **tb_break** command:

        ```
        /usr/sbin/rsct/bin/tb_break -v -u -t SCSIPR DeviceInfo_device_specification_for_this_system
        ```

        This command releases the disk device.

For example, to issue a reserve command to a disk on Linux, on the Power Systems or System x hardware platforms, run this **tb_break** command:

```
tb_break -l -t SCSI "RDAC.WWN=600a0b80000cf57b00000042474304bc"
```

The following output is displayed:

```
Initializing SCSI tie-breaker (RDAC.WWN=600a0b80000cf57b00000042474304bc)
WWN (600a0b80000cf57b00000042474304bc) is found at /proc/mpp/C105FastT/virtualLun4
Found: WWN=600a0b80000cf57b00000042474304bc HOST=2 CHAN=0 ID=0 LUN=4
Reserving tie-breaker (RDAC.WWN=600a0b80000cf57b00000042474304bc)
scsi_reserve(/dev/sg4) is granted, status=0
tb_reserve status 0 (errno=0)
```

To release the reservation, run this **tb_break** command:

```
tb_break -u -t SCSI "RDAC.WWN=600a0b80000cf57b00000042474304bc"
```

The following output is displayed:

```
Initializing SCSI tie-breaker (RDAC.WWN=600a0b80000cf57b00000042474304bc)
WWN (600a0b80000cf57b00000042474304bc) is found at /proc/mpp/C105FastT/virtualLun4
Found: WWN=600a0b80000cf57b00000042474304bc HOST=2 CHAN=0 ID=0 LUN=4
Releasing tie-breaker (RDAC.WWN=600a0b80000cf57b00000042474304bc)
tb_release status 0 (errno=0)
```

If the disk is not capable of accepting SCSI-2 reservations, you receive the following error:

```
tb_break -l -t SCSI "WWID=6005076303ffc4d20000000000001172"
Initializing SCSI tie-breaker (WWID=6005076303ffc4d20000000000001172)
tb_set_error (errid=-1 msg_num=106)error in tb_init(). status=-1
```

Explicitly resolving a tie when the active tiebreaker type is Operator:

When the active tiebreaker is the predefined tiebreaker Operator or a tiebreaker whose persistent attribute Type is Operator, then the configuration resource manager does not automatically resolve tie situations.

If domain partitioning occurs with a subdomain containing exactly half the defined nodes (or if exactly half of the domain's defined nodes become inaccessible), the configuration manager sets the OpQuorumState dynamic attribute of the PeerDomain resource to 1 (PendingQuorum). Operational quorum is not granted until either the network is repaired, failing nodes are brought online, or you explicitly break the tie by issuing the ResolveOpQuorumTie action of the IBM.PeerNode resource class.

To resolve a tie situation using the ResolveOpQuorumTie action, you must invoke the action on a node of each active subdomain. The single input parameter to this action is an integer that indicates whether the subdomain in which the action is invoked is denied (0) or granted (1) or ownership of the tiebreaker.

When explicitly resolving a tie between subdomains, in order to avoid corruption of shared data, first deny ownership of the tiebreaker to the appropriate subdomain. Once you have denied ownership of the tiebreaker to the appropriate subdomain, you can safely grant ownership of the tiebreaker to the subdomain that you want to have operational quorum.

To deny ownership of the Operator tiebreaker to a subdomain, invoke the following action on a node of that subdomain.

```
runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=0
```

Denying ownership of the tiebreaker to a subdomain causes the configuration manager to set the OpQuorumState dynamic attribute of the PeerDomain resource to 2 (NoQuorum). The subdomain loses quorum, which in turn might cause the critical resource protection method to be invoked on any nodes that have critical resources active.

To grant ownership of the Operator tiebreaker to a subdomain, invoke the following action on a node of that subdomain.

```
runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=1
```

Granting ownership of the tiebreaker to a subdomain causes the configuration manager to set the OpQuorumState dynamic attribute of the PeerDomain resource to 0 (HasQuorum). The subdomain has operational quorum and so becomes the peer domain.

Related tasks:

"Setting the critical resource protection method for a peer domain or a node in a peer domain" on page 86

You can set the critical resource protection method for a peer domain by setting the **CritRsrcProtMethod** persistent attribute of the **IBM.PeerNode** resource class.

Administering shared secret keys

Shared secret keys can be used by Topology Services, Group Service, and RMC to authenticate control messages sent between nodes within a peer domain.

Normally, the configuration resource manager automatically manages an active shared secret key within the peer domain. However, you can perform several tasks to administer the use of shared secret keys, such as:

- List the active key type and refresh interval
- Enable or disable the use of a shared secret key
- Change the key type or refresh interval for a shared secret key
- Manually refresh the shared secret key

These tasks can only be performed while the peer domain is online. In addition, changing the key type and refresh interval can only be done while the peer domain is online and has quorum.

Listing the key type and refresh interval for a shared secret key

Two persistent resource attributes of the IBM.RSCTParameters resource class control the function of the shared secret key for a peer domain.

The CSSKType attribute contains the key type and the CSSKRefreshInterval attribute contains the refresh interval.

Use the **lsrsrc** command to display the key type and refresh interval for a shared secret key.

```
lsrsrc -c IBM.RSCTParameters CSSKType CSSKRefreshInterval
```

Enabling use of a shared secret key

To enable the use of a shared secret key for an online peer domain that does not currently use one, issue the **chrsrc** command.

To enable the use of a shared secret key for an online peer domain that does not currently use one, issue the **chrsrc** command and specify the key type that you want to use:

```
chrsrc -c IBM.RSCTParameters CSSKType=key_type
```

For *key_type*, specify one of the valid key types:

CSSKTYPE_DES_MD5
: DES encryption using a 56-bit key

CSSKTYPE_3DES_MD5
: Triple DES encryption using a 168-bit key

CSSKTYPE_AES256_MD5
: AES encryption using a 256-bit key

The key types are further described under the **mkrpdomain** command.

When the **chrsrc** command completes, a key value of the specified key type will be generated and propagated for use across the peer domain and the refresh interval for the key will be set to the default of one day (86400 seconds). Topology Services, Group Services, and RMC control messages will be signed for authentication using the shared secret key.

For more information about the **chrsrc** and **mkrpdomain** commands, see their online man pages or to *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Disabling use of a shared secret key

To disable the use of a shared secret key for an online peer domain that is currently using one, issue the **chrsrc** command.

To disable the use of a shared secret key for an online peer domain that is currently using one, issue the **chrsrc** command and specify a key type of *CSSKTYPE_None*:

```
chrsrc -c IBM.RSCTParameters CSSKType=CSSKTYPE_None
```

When the **chrsrc** command completes, Topology Services, Group Services, and RMC will no longer be using a shared secret key to authenticate their message traffic.

For more information about the **chrsrc** command, see its online man page or to *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Changing the key type for the shared secret key

To change the key type of the shared secret key for an online peer domain, issue the **chrsrc** command.

To change the key type of the shared secret key for an online peer domain, issue the **chrsrc** command and specify the desired new key type:

```
chrsrc -c IBM.RSCTParameters CSSKType=new_key_type
```

The value you specify for *new_key_type* must be one of the valid key types described in "Enabling use of a shared secret key" on page 103

When the **chrsrc** command completes, a key value of the specified new key type will be generated and propagated for use across the peer domain and the refresh interval for the new key will be set to the current key refresh interval specified in the CSSKRefreshInterval attribute of the IBM.RSCTParameters resource class. Topology Services, Group Services, and RMC control messages will be signed for authentication using the new shared secret key.

For more information about the **chrsrc** command, see its online man page or to *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Changing the refresh interval for the shared secret key

To change the refresh interval for the shared secret key for an online peer domain, issue the **chrsrc** command.

To change the refresh interval for the shared secret key for an online peer domain, issue the **chrsrc** command and specify a new refresh interval, in seconds:

```
chrsrc -c IBM.RSCTParameters CSSKRefreshInterval=seconds
```

To disable automatic key refreshes, specify a value of 0 seconds for the refresh interval.

For more information about the **chrsrc** command, see its online man page or to *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Manually refreshing the shared secret key

To manually refresh the shared secret key for an online peer domain, issue the **runact** command.

To manually refresh the shared secret key for an online peer domain, issue the **runact** command and specify the UpdateKey parameter:

```
runact -c IBM.RSCTParameters UpdateKey
```

This command will refresh the value of the currently active shared secret key and restart the refresh interval timer.

For more information about the **chrsrc** command, see its online man page or see *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Monitoring critical file system

The **IBM.ConfigRM** resource class monitors the root file system similarly as the **IBM.StorageRM** resource class monitors file systems.

If a critical resource is active, the **IBM.ConfigRM** resource class creates a hidden file called **.rsct-health-check-DO_NOT_DELETE** to monitor the root file system. It writes to this file every 60 seconds to make sure that the file system is writeable. If the **IBM.ConfigRM** resource class is unable to write to the file, it reboots the node, assuming the **CritRsrcProtMethod** attribute for the **IBM.PeerNode** class is set to 0 or 1. Otherwise, if the **CritRsrcProtMethod** attribute for the **IBM.PeerNode** class is set to another value, the setting dictates the action taken by the node.

crit_fs.conf file

The **crit_fs.conf** file is a global system configuration file that is in the **/var/ct/cfg** directory. The **crit_fs.conf** file is used to do the following tasks:

- To determine the **MonitorInterval** value. The **MonitorInterval** value is the amount of time (in seconds) between the monitoring activities.

  Generally, the root file system is monitored automatically, but if the **crit_fs.conf** file is used, the **MonitorInterval** value monitors the root file system. If the **MonitorInterval** value is not specified, the default interval is 60 seconds.
- To determine the **MonitorTimeout** value. The **MonitorTimeout** value is the amount of time (in seconds) that needs to be elapsed before taking any action after the last failed root file system monitoring.

  Note: The **MonitorTimeout** value must be larger than the **MonitorInterval** value.
- To determine the **MonitorType** value. The **MonitorType** attribute decides whether an action needs to be taken if there are critical resources or not. It has three values:

Table 29. MonitorType values

Value	Description
0	Takes no action.
1	Monitors when a critical resource is present. This value is the default setting.
2	Monitors at all times, no matter if a critical resource is present or not.

When the **MonitorType** variable has values of 1 or 2, and a critical resource is present, if the monitoring fails, the node restarts assuming the default values of the **CritRsrcProtMethod** attribute in the **IBM.PeerNode** class are set.

If the **IBM.PeerDomain** class is not running, that is, if the **CritRsrcProtMethod** attribute of the **IBM.PeerNode** class is not used, the RSCT services are restarted.

The contents of the **/var/ct/cfg/crit_fs.conf** file can be similar to the following example:

```
MonitorTimeout = 90  # in seconds - to take the CritRsrcProtMethod attribute after unsuccessful monitoring.
MonitorInterval = 30 # in seconds - interval between monitoring activity.
MonitorType = 1      # monitors the root file system only when a critical resource is present.
```

The **IBM.ConfigRM** resource class reads the **crit_fs.conf** file:

- After the **refrsrc IBM.PeerDomain** command is run.
- If the **ConfigRMd** daemon is restarted.
- At domain startup.

Note: If the **crit_fs.conf** file has its **MonitorType** value changed to 0, then the **IBM.ConfigRM** resource class reads the **crit_fs.conf** file only if the **refrsrc IBM.PeerDomain** command is run. Removing and creating a domain does not change the behavior of the daemon.

Understanding RMC and resource managers

An understanding of RMC and resource manager concepts is essential before performing resource management tasks.

Before performing resource management tasks, understand the following concepts:

- How the RMC subsystem provides a generic way to represent, and manage various physical and logical system entities.
- How a set of resource managers map information about specific entities to RMC's abstractions.
- The representational components of RMC's generic framework. These include resources (the physical or logical system entities represented), attributes (characteristics of resources), and resource classes (sets of resources with common attributes).
- The resource managing capabilities of RMC and the resource managers.
- The monitoring capabilities of RMC and the resource managers (described in more detail later in "Basic resource monitoring" on page 136 and "Advanced resource monitoring" on page 152).
- How RMC implements authorization (described in more detail later in "Managing user access to resources using RMC ACL files" on page 126).
- Differences between using RMC on a single node versus a cluster.

Resource monitoring and control

The resource monitoring and control (RMC) subsystem is a generalized framework for managing, monitoring, and manipulating resources (physical or logical system entities).

RMC runs as a daemon process on individual machines, and, therefore, is scalable. You can use it to manage and monitor the resources of a single machine, or you can use it to manage and monitor the resources of a cluster's peer domain or management domain. In a peer domain or management domain, the RMC daemons on the various nodes work together to enable you to manage and monitor the domain's resources.

A resource

A *resource* is the fundamental concept of RMC's architecture. It refers to an instance of a physical or logical entity that provides services to some other component of the system. The term resource is used very broadly to refer to software as well as hardware entities. For example, a resource could be a particular file system or a particular host machine.

A resource class

A *resource class* is a set of resources of the same type. For example, while a resource might be a particular file system or particular host machine, a resource class would be the set of file systems, or the set of host machines. A resource class defines the common characteristics that instances of the resource class can have; for example, all file systems will have identifying characteristics (such as a name), as well as changing characteristics (such as whether or not it is mounted). Each individual resource instance of the resource class will then define what its particular characteristic values are (for example, this file system is named "/var", and it is currently a mounted file system).

Resource attributes

A resource *attribute* describes some characteristic of a resource. If the resource represents a host machine, its attributes would identify such information as the host name, size of its physical memory, machine type, and so on. There are two types of resource attributes — *persistent attributes* and *dynamic attributes*.

Persistent attributes:

The attributes of a host machine, such as its host name, physical memory size, and machine type, are examples of *persistent attributes*—they describe enduring characteristics of the resource. While you could change the host name or increase the size of its physical memory, these characteristics are, in general, stable and unchanging.

Persistent attributes are useful for identifying particular resources of a resource class. There are many commands for directly or indirectly manipulating resources. Persistent attributes enable you to easily identify an individual resource or set of resources of a resource class that you want to manipulate. For example, the **lsrsrc** command lists resource information. By default, this command will list the information for all resources of the class. However, you can filter the command using persistent attribute values. In a cluster, this ability would enable you to list information about a particular host machine (by filtering using the host's name) or a group of host machines of the same type (by filtering according to the machine type). Although listing resources can be considered a basic task, this ability to identify resources by their attributes and isolate command actions to a single resource or subset of resources is available on many of the more advanced commands. This ability gives you increased flexibility and power in managing resources.

Dynamic attributes:

Dynamic attributes represent changing characteristics of the resource. Dynamic attributes of a host resource, for example, would identify such things as the average number of processes that are waiting in the run queue, processor idle time, and the number of users who are currently logged on.

A dynamic attribute is further classified by its *variable type*, which is one of the following:

- **Counter**

  This variable type represents a rate that is the difference between the current raw value and the previous raw value, divided by the time, in seconds, between the sampling of the raw values. The raw values are obtained from the underlying resource. The calculated value is the value of the attribute. For example, the following expression generates an event when the page-in rate exceeds 1000 per second:

  ```
  VMPgInRate > 1000
  ```

  The raw value can also be referenced in an expression using the **@R** suffix:

  ```
  (VMPgInRate@R - VMPgInRate@RP) > 2 * (VMPgOutRate@R - VMPgOutRate@RP)
  ```

  This expression generates an event notification if, during any sampling interval, the number of page-ins exceeds twice the number of page-outs. Note that the **P** suffix indicates the previous value. The RMC subsystem maintains the *N*th and *N*th -1 value of each monitored attribute.

- **Quantity**

  This variable type represents a value that fluctuates over time, and typically represents a level.

- **Quantum**

  This variable type signifies a change. A **Quantum** attribute has no value associated with it. Such an attribute is regarded as a Boolean that is always True. A resource manager asserts a **Quantum** attribute by reporting a new "value". The act of reporting generates the event notification. For a **Quantum** attribute, the expression consists solely of the attribute name:

  ```
  ConfigChanged
  ```

- **State**

This variable type represents a value that fluctuates over time. Every change in value of a **State** attribute might be meaningful. Therefore, a resource manager cannot sample such an attribute; every change in value must be reported as it occurs. A State attribute has no reporting interval. For example:

```
OpState == 1 && OpState@P == 5
```

This expression generates an event notification when a resource moves from the pending online state to the online state. A missed value change would prevent the proper generation of event notifications. A persistent attribute, while it does not have a variable type, is regarded as a **State** attribute when it is used in an expression.

A **Counter** or **Quantity** attribute has its value sampled by the resource manager without the loss of meaningful information. The sampling rate, or reporting interval, is part of the class definition. The **lsrsrcdef -ad** *resource_class_name* command can be used to obtain the reporting intervals of dynamic attributes. For example, **PercentTotUsed** and **PercentINodeUsed** are **Quantity** attributes. Every change in the number of disk blocks or i-nodes allocated need not be reported in order to generate interesting events.

Monitoring is the periodic evaluation of an event expression. This period may be regular or irregular, depending on the variable type of the attributes within the expression. If the expression specifies a single named attribute and that is a **Counter** or **Quantity** attribute, the resource manager provides a new value to the RMC subsystem every reporting interval seconds and then the expression is evaluated.

If the single named attribute is a **State** attribute, the resource manager typically provides a new value whenever there is a change from a prior value and then the expression is evaluated. The period between expression evaluations, in this case, is indeterminate from the point of view of the RMC client. Note that, depending on the implementation of the resource manager and the nature of the resource itself, **State** attribute values might be reported even if there is no change in value. By definition, however, **State** attribute values are reported when there is a change in value.

If the single named attribute is a **Quantum** attribute, the resource manager asserts the attribute whenever there is a change in the state of the resource or resource class, as represented by the attribute, and then the expression is evaluated. For example, the **ConfigChanged** class dynamic attribute is asserted whenever a new resource is created.

When a resource class defines a set of dynamic attributes of variable type **State**, the resource manager might report their new values at regular intervals. In this case, the reporting interval is not defined in the class definition; rather, the interval is typically defined by the value of a resource persistent attribute. The **IBM.Sensor** and **IBM.MicroSensor** resources support this capability. Upon completion of one of the register event commands, monitoring begins for each specified resource, or the resource class itself, as specified by the event expression. The first expression evaluation occurs, per monitored object, once every attribute named in the expression has a reported value. This is immediately, if all the attributes are already being monitored as a result of earlier register event commands. Subsequently, the expression is evaluated each time any of the specified attributes has a new value, as reported by the resource manager. It may be the case that the values for multiple attributes are reported together, or individually or some mix thereof. How values are reported is a function of the reporting intervals, if any, and the resource manager implementation.

Dynamic attributes are useful in monitoring your system for conditions of interest. As described in “Basic resource monitoring” on page 136 and “Advanced resource monitoring” on page 152, you can monitor events of interest (called *conditions*) and have the RMC system react in particular ways (called *responses*) if the event occurs. The conditions are logical expressions based on the value of one or more attributes. For example, there is a resource class used to represent file systems. You could create a condition to monitor the file systems and trigger a response if any of them become more than 90 percent full. The percentage of space used by a file system is one of its dynamic attribute values. It usually does not make sense to monitor persistent attribute values, since they are generally unchanging. For example, if you wanted to monitor a file system, it would not make sense to monitor based on the file system name (a persistent

attribute). However, you may want to use this persistent attribute to identify a particular file system resource to monitor. Instead of monitoring all file systems, you could use this persistent attribute value to identify one particular file system to monitor.

An action

An *action* is an operation, specific to a resource class, that can be performed on either a resource of the resource class, or on the resource class itself. You can use the **lsactdef** command to display a list of the action definitions of a resource class, including the input that can be provided to, and the output that can be returned from, each action. To actually run an action, you can use the **runact** command.

For more information on the **lsactdef** and **runact** commands, see their online man pages. For detailed information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

A resource manager

A resource manager is a daemon process that provides the interface between RMC and actual physical or logical entities.

It is important to understand that although RMC provides the basic abstractions (resource classes, resources, and attributes) for representing physical or logical entities, it does not itself represent any actual entities. A resource manager maps actual entities to RMC's abstractions.

Each resource manager represents a specific set of administrative tasks or system features. The resource manager identifies the key physical or logical entity types related to that set of administrative tasks or system features, and defines resource classes to represent those entity types.

For example, the host resource manager contains a set of resource classes for representing aspects of an individual host machine. It defines resource classes to represent the following resources:

- individual machines (**IBM.Host**)
- paging devices (**IBM.PagingDevice**)
- physical volumes (**IBM.PhysicalVolume**)
- processors (**IBM.Processor**)
- a host's identifier token (**IBM.HostPublic**)
- programs running on the host (**IBM.Program**)
- each type of adapter supported by the host, including ATM adapters (**IBM.ATMDevice**), Ethernet adapters (**IBM.EthernetDevice**), FDDI adapters (**IBM.FDDIDevice**), and token-ring adapters (**IBM.TokenRingDevice**)

The resource class definitions describe the persistent and dynamic attributes that individual resource instances of that class can or must define. For example, the **Host** resource class defines persistent attributes such as **Name** (the name of the host machine), **RealMemSize** (the size of physical memory in bytes), and **OsVersion** (the version of the operating system or kernel running on the host machine). It defines dynamic attributes such as **PctTotalTimeIdle** (system-wide percentage of time that processors are idle), **NumUsers** (number of users currently logged on to the system), and **UpTime** (the number of seconds since the system was last booted).

A resource manager also determines how individual resources of each class are identified. Although you can use the **mkrsrc** command to explicitly define a resource, this is often not necessary, since resources may be automatically harvested by the resource manager. For example, there is a resource manager used to represent file systems. This resource manager harvests (gathers information on) existing file systems to create resources representing those file systems. It will periodically repeat this harvesting to so that its resources are still representative of the actual file systems available. In addition to harvesting, resources may be created implicitly by other commands. For example, the Host resource manager has a Program

resource class that represents programs running on the host. If you were to create a monitoring condition (described in "Creating a condition" on page 157) referring to a particular program, a Program resource representing the program is created implicitly.

Another job of a resource manager to determine the dynamic attribute values of its resources. Since dynamic attributes represent changing characteristics of a resource, the resource manager will periodically poll the actual resources to determine the dynamic attribute values. This is essential to enable resource monitoring (described in "Basic resource monitoring" on page 136 and "Advanced resource monitoring" on page 152) where conditions used to trigger responses are logical expressions based on the value of an attribute. It is the periodic polling of resources that enables the event driven condition/response behavior of resource monitoring.

While some resource managers represent system features (such as individual host machines of a cluster, or file systems) others represent resources related to a specific administrative task (such as peer domain configuration, or resource monitoring). Since the purpose of such a resource manager is to provide administrative function, it will provide a command-line interface for performing the administrative tasks. For example, the Configuration resource manager (described in "Administering an RSCT peer domain" on page 35) provides commands for creating creating a peer domain, adding nodes to the domain, taking the domain offline, and so on.

Each resource manager has a startup mode that determines when the RMC subsystem will start it. The startup mode types are:

auto-start
: The resource manager is started when the RMC subsystem is started.

conditional auto-start
: The resource manager is started when the RMC subsystem is started and when some condition is satisfied.

on-demand
: The resource manager is started when one of its resources is first referenced.

online auto-start
: The resource manager is started when the node becomes online in a peer domain.

The startup mode for each RSCT resource manager is indicated in Table Table 30 on page 111 in "Resource managers provided with RSCT."

The RMC subsystem can also consider certain resource managers to be idle when they have no work to do for a specified period of time. To conserve system resources, RMC can then gracefully shut down those idle resource managers. RMC currently uses a value of ten minutes to decide that a resource manager is idle. Whether or not a resource manager is subject to automatic shut down when it is idle is also indicated in Table Table 30 on page 111 in "Resource managers provided with RSCT."

Resource managers provided with RSCT

Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

Table 30 on page 111 describes the resource managers that are provided as part of RSCT. Certain cluster licensed programs, such as Tivoli System Automation, provide additional resource managers. Some resource managers are not available on all platforms.

Table 30. Resource managers provided with RSCT

Resource manager	Description	Startup mode	Automatic shutdown when idle?
Audit log resource manager	Provides a system-wide facility for recording information about the system's operation. Subsystem components use this resource manager to log information about their actions and errors. For example, the event response resource manager, which provides resource monitoring functions, uses the audit log resource manager to log information about condition events and responses that are occurring. A command-line interface to the audit log resource manager enables you to list and remove records from an audit log.	On-demand	Yes
CIM resource manager	Enables you to use RMC on AIX or Linux to query system configuration through Common Information Model (CIM) classes. The CIM resource manager provides a command called **mkcimreg** that enables you to register CIM classes with RMC. Once registered, you can query a CIM property or association, or monitor a CIM property, through RMC.	On-demand	Yes
Configuration resource manager	Provides the ability, through its command-line interface, to create and administer a peer domain, which is a cluster of nodes that is configured for high availability.	Conditional auto-start (started if the node is defined in a peer domain; otherwise, started on-demand)	No
Event response resource manager	Provides resource monitoring, that is, the ability to act in response to conditions occurring in the system. Its command-line interface enables you to associate conditions with responses, start and stop condition monitoring, and so on.	Conditional auto-start (started if one or more **IBM.Condition** resources are previously activated; otherwise, started on-demand)	Yes
File system resource manager	Provides a resource class to represent file systems. This resource manager has no user interface. Instead, you interact with it indirectly when you monitor its resource attributes using the event response resource manager. **Note:** This resource manager is available on AIX and Linux only.	On-demand	Yes
Host resource manager	Provides resource classes to represent an individual machine, including its paging devices, physical volumes, and processors. This resource manager has no user interface. Instead, you interact with it indirectly when you monitor its resource attributes by using the event response resource manager.	On-demand	Yes
Least-privilege resource manager	Controls access to root commands or scripts and controls the local or remote execution of those commands or scripts. The least-privilege (LP) resource manager provides a resource class and a command-line interface that enables you to define, manage, and monitor root commands and scripts as LP resources.	On-demand	Yes
Management domain resource manager	Provides resource classes to support configuration of an RSCT management domain. A management domain is a set of nodes with resources that can be managed and monitored from one of the nodes, which is called the management control point (MCP). All other nodes are considered to be *managed nodes.* This resource manager is intended to be used by the Extreme Cloud Administration Toolkit (xCAT). The resources supported by this resource manager are defined, modified, and removed by using the standard RMC commands **mkrsrc**, **chrsrc**, and **rmrsrc**. However, even without xCAT, you can create a management domain by using these commands.	On-demand	Yes

Table 30. Resource managers provided with RSCT (continued)

Resource manager	Description	Startup mode	Automatic shutdown when idle?
Microsensor resource manager	Provides a way to extend the monitoring capabilities of the system by enabling you to create user-defined attributes for monitoring. Extending the system in this way involves creating a microsensor. A microsensor is a dynamically loaded shared library which provides C-based functions that the RMC subsystem runs (at specified intervals and/or when you explicitly request for it to be run to: • obtain control information about the microsensor • stop the monitoring of some user-defined values • start the monitoring of some user-defined values • retrieve the user-defined values • notify the library that it is no longer needed The microsensor is essentially a resource that you add to the microsensor resource class of the microsensor resource manager. The values returned by the functions are dynamic attributes of that resource. Using the event response resource manager commands, you can then create a condition to monitor any of the attributes you defined. The microsensor resource manager provides a command-line interface for creating, changing, listing, and removing sensors.	On-demand	Yes
Sensor resource manager	Provides a way to extend the monitoring capabilities of the system by enabling you to create a single user-defined attribute for monitoring. Extending the system in this way involves creating a *sensor*. A sensor is merely a command that the RMC subsystem runs (at specified intervals and/or when you explicit request for it to be run) to retrieve one or more user-defined values. The sensor is essentially a resource that you add to the Sensor resource class of the Sensor resource manager. The values returned by the script are dynamic attributes of that resource. Using the event response resource manager commands, you can then create a condition to monitor any of the attributes you defined. The sensor resource manager provides a command-line interface for creating, changing, listing, and removing sensors.	On-demand	Yes
Storage resource manager	Provides management and data protection capabilities for shared storage resources within a RSCT peer domain. The storage resource manager provides the interface between RMC and the physical and logical storage entities within the peer domain by mapping these entities to instances of the resource classes it provides. Running as a daemon process on each node in the peer domain, the storage resource manager collects information about locally attached physical disks (and related storage entities) and maps it to resource class instances. These separate views of the storage resources from each individual node are then collected together to provide the Storage resource manager with a global view of the storage resources. **Note:** This resource manager is not available on Windows.	On-demand	No

Related concepts:

"Resource classes defined by the audit log resource manager" on page 113
Generally, you will not need to manipulate resources of these classes directly. Instead, you would manipulate the audit log using the **lsaudrec** and **rmaudrec** commands.

Related tasks:

"Using the audit log to track monitoring activity" on page 145
When you are monitoring a condition or compound condition, be aware that any linked response actions will be executed in the background by daemons.

Related information:

"Querying and monitoring CIM properties and associations" on page 184
The Common Information Model (CIM) is a data model for organizing computer hardware and software resources into a common object-oriented class hierarchy. Developed and maintained by the Distributed Management Task Force (DMTF), CIM is a conceptual model for extracting management information. In this way, it is similar to the RMC data model.

"Administering an RSCT peer domain" on page 35
To achieve high availability, you can configure a cluster of nodes into an RSCT peer domain.

"Basic resource monitoring" on page 136
You can use Event Response Resource Manager commands to monitor your system of cluster domains.

"Advanced resource monitoring" on page 152
Many predefined conditions and responses are provided by the various resource managers on your system. These predefined conditions and responses are provided as an administrative convenience.

Resource classes defined by the audit log resource manager:

Generally, you will not need to manipulate resources of these classes directly. Instead, you would manipulate the audit log using the **lsaudrec** and **rmaudrec** commands.

Table 31 describes the resource classes defined by the audit log resource manager. In general, you will not need to manipulate resources of these classes directly. Instead, you would manipulate the audit log using the lsaudrec command (as described in "Using the audit log to track monitoring activity" on page 145) and **rmaudrec** command (as described in "Deleting entries from the audit log using the rmaudrec command" on page 150). One instance where you would manipulate a resource of the IBM.AuditLog class directly would be to set the RetentionPeriod or MaxSize attribute values (as described in "Deleting entries from the audit log using the IBM.AuditLog resource's RetentionPeriod and MaxSize attributes" on page 151).

Table 31. Resource classes defined by the audit log resource manager

Resource class	Description
IBM.AuditLog	Each resource of this class represents a subsystem that can add records to the audit log.
IBM.AuditLogTemplate	Each resource of this class represents a template for an audit log record.

For information on listing attribute values or attribute definitions for the resource classes (or resource instances of the resource classes) listed in the preceding table, see "Listing resource information" on page 130.

Related concepts:

"Resource managers provided with RSCT" on page 110
Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

Resource classes defined by the CIM resource manager:

The CIM resource manager does not provide a default set of resource classes. Instead, the CIM resource manager enables you to use RMC to query or monitor system configuration through Common Information Model (CIM) classes. CIM is a data model, similar to the RMC data model, for organizing computer hardware and software resources into a common object-oriented class hierarchy.

The CIM resource manager provides a command that enables you to register CIM properties with RMC. The CIM classes are mapped to new RMC resource classes. The RMC resource class name will be a concatenation of the namespace and the CIM class name — for example *cimv2.Linux_ComputerSystem*. All registered CIM classes are placed in the root/cimv2 namespace.

For more information on the CIM resource manager, see “Querying and monitoring CIM properties and associations” on page 184.

Resource classes defined by the configuration resource manager:

In general, you do not need to manipulate resources of these classes directly. Instead, use the configuration resource manager commands.

Table 32 describes the resource classes defined by the configuration resource manager. In general, you do not need to manipulate resources of these classes directly. Instead, use the configuration resource manager commands. However, you might need to:

- Modify attributes of the **RSCTParameters** class.
- Modify the **CritRsrcProtMethod** attribute of the **IBM.PeerNode** class or a resource instance of the **IBM.PeerNode** class.
- Modify the **OpQuorumOverride** attribute of the **IBM.PeerNode** class.
- Modify the **OpQuorumTieBreaker** attribute of the **IBM.PeerNode** class.
- Modify the **IsPreferredGSGL** attribute of the **IBM.PeerNode** class.

Table 32. Resource classes defined by the configuration resource manager

Resource class	Description
IBM.CommunicationGroup	Each resource of this class represents a communication resource upon which liveness checks (Topology Services heartbeat) is performed.
IBM.HeartbeatInterface	Each resource of this class represents a heartbeat interface that exists in the peer domain.
IBM.NetworkInterface	Each resource of this class represents an IP network interface that exists in the peer domain.
IBM.PeerDomain	Each resource of this class represents an RSCT peer domain in which a particular node is defined. Each node has its own IBM.PeerDomain resource class, with each instance of the resource class, representing an RSCT peer domain in which the node is defined. The number of instances of this resource class, therefore, indicates the number of peer domains in which the node is defined. For the **DomainType** resource persistent attribute, `DomainType = 1` indicates that RPD is created in the CAA environment by using **mkrpdomain** with the **–C** option. `DomainType = 0` indicates that the RPD is created by using the **mkrpdomain** command without the **–C** option.
IBM.PeerNode	Each resource of this class represents a node defined in the peer domain. A node is here defined as an instance of an operating system, and is not necessarily tied to hardware boundaries.
IBM.RSCTParameters	Represents operational characteristics of RSCT subsystems.
IBM.TieBreaker	Each resource of this class represents a tiebreaker. A tiebreaker is used, when domain partitioning results in two subdomains, each containing exactly one-half of the nodes, to determine which subdomain has an operational quorum. In **QuorumLess** mode, a tiebreaker is always used to determine the subdomain that has an operational quorum.

Related tasks:

“Listing resource information” on page 130
The **lsrsrc** and **lsrsrcdef** commands enable you to list information about the resources available on your system or cluster.

Related information:

“Administering an RSCT peer domain” on page 35
To achieve high availability, you can configure a cluster of nodes into an RSCT peer domain.

Resource classes defined by the event response resource manager:

Generally, you will not need to manipulate resources of these classes directly. Instead you would use the event response resource manager commands.

Table 33 describes the resource classes defined by the event response resource manager. In general, you will not need to manipulate resources of these classes directly. Instead you would use the event response resource manager commands described in "Basic resource monitoring" on page 136 and "Advanced resource monitoring" on page 152.

Table 33. Resource classes defined by the event response resource manager

Resource class	Description
IBM.Association	Each resource of this class represents a condition/response association. Such an association enables the RMC subsystem to trigger one or more response actions when a particular condition occurs.
IBM.CompoundCondition	Each resource of this class represents a compound condition (an event defined by an expression on one or more condition resources (of the IBM.Condition resource class) that should trigger a response).
IBM.Condition	Each resource of this class represents a condition (an event that should trigger a response).
IBM.EventResponse	Each resource of this class represents a response (one or more actions the system can take when a condition event occurs).

For information on listing attribute values or attribute definitions for the resource classes (or resource instances of the resource classes) listed in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the file system resource manager:

In general, you will not need to manipulate resources of this class directly. However, you may want to monitor file systems using the dynamic attributes of IBM.FileSystem resources.

Table 34 describes the resource class defined by the file system resource manager. Monitoring is described in "Basic resource monitoring" on page 136 and "Advanced resource monitoring" on page 152.

Table 34. Resource classes defined by the file system resource manager

Resource class	Description
IBM.FileSystem	Each resource of this class represents a file system.

For information on listing attribute values or attribute definitions for the resource class (or resource instances of the resource classes) shown in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the host resource manager:

In general, you will not need to manipulate resources of these classes directly. However, you may want to monitor host machines using resource dynamic attributes of the various resource classes.

Table 35 on page 116 describes the resource classes defined by the host resource manager. Monitoring is described in "Basic resource monitoring" on page 136 and "Advanced resource monitoring" on page 152.

Table 35. Resource classes defined by the host resource manager

Resource class	Description	Resource classes available only on AIX	Resource classes available only on AIX and Linux
IBM.ATMDevice	Each resource of this class represents an ATM adapter installed on the host.	X	
IBM.EthernetDevice	Each resource of this class represents an Ethernet adapter installed on the host.		X
IBM.FDDIDevice	Each resource of this class represents a FDDI adapter installed on the host.	X	
IBM.Host	Each resource of this class represents a host machine that is running a single copy of an operating system.		X
IBM.HostPublic	Each resource of this class represents the host's public key.		
IBM.PagingDevice	Each resource of this class represents a device that is used by the operating system for paging.	X	
IBM.PhysicalVolume	Each resource of this class represents a physical volume.	X	
IBM.Processor	Each resource of this class represents a processor.	X	
IBM.Program	Each resource of this class represents a program that is running on the host.		X
IBM.TokenRingDevice	Each resource of this class represents a token-ring device installed on the host.	X	

For information on listing attribute values or attribute definitions for the resource classes (or resource instances of the resource classes) listed in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the least-privilege resource manager:

In general, you will not need to manipulate resources of this class directly. Instead, you will use the least-privilege resource manager commands.

Table 36 describes the resource class defined by the least-privilege resource manager. In general, you will not need to manipulate resources of this class directly. Instead, you will use the least-privilege resource manager commands described in "Controlling access to root commands and scripts" on page 214.

Table 36. Resource classes defined by the least-privilege resource manager

Resource class	Description
IBM.LPCommands	Each resource of this class represents a root command or script that only authorized users may run.

For information on listing attribute values or attribute definitions for the resource class (or resource instances of the resource classes) shown in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the management domain resource manager:

The resource classes of the management domain resource manager represent the various nodes within a management domain and the network interfaces of the managed nodes. In addition, there is one resource class used by the resource manager to automatically exchange the public key of each managed node with its Management Control Point (MCP). The RMC commands **mkrsrc**, **chrsrc**, and **rmrsrc** are used to manage the resources of the **IBM.MngNode** and **IBM.MCP** classes. Resources of the **IBM.MngNodeNetIF** class are automatically created, modified, and removed by the resource manager as

resources of the **IBM.MngNode class** are created, modified, and removed. The **IBM.PublicKeyExchange** class is used by the resource manager to exchange the public keys of an MCP and a managed node.

Table 37 describes the resource class defined by the management domain resource manager.

Table 37. Resource classes defined by the management domain resource manager

Resource class	Description
IBM.MCP	This resource manager on a managed node configures a management control point (MCP) to the managed node. Exchange of host public keys is automatically performed as **IBM.MCP** resources are created on managed nodes, using the **IBM.PublicKeyExchange** class. Default ACLs are created on the managed nodes for the root ID of the MCP at the same time. Resources of this class can be defined, modified, and removed using the RMC commands **mkrsrc**, **chrsrc**, and **rmrsrc**.
IBM.MngNode	This resource class on the MCP configures a managed node to the management domain. The first such instance designates the system as the MCP of a management domain. Resources of this class can be defined, modified, and removed using the RMC commands **mkrsrc**, **chrsrc**, and **rmrsrc**.
IBM.MngNodeNetIF	Resources of this class are created and defined automatically as associated **IBM.MngNode** resources are defined, modified, and removed, one per IP address configured to a managed node.
IBM.PublicKeyExchange	Normally, a public key exchange (PKE) protocol is automatically initiated between a managed node and its MCP when an **IBM.MCP** resource is successfully created on the managed node. This requires that the **IBM.MngNode** resources be created first on the MCP. The protocol is also initiated whenever the **refrsrc IBM.MCP** command is run on a managed node, for all **IBM.MCP** resources defined on the node at that time. This resource class does not support resource instances.
IBM.RMCCtrl	The single resource of this class is used to set the heartbeating parameters for the RMC.

For information on listing attribute values or attribute definitions for the resource class (or resource instances of the resource classes) shown in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the microsensor resource manager:

The microsensor resource manager provides a way to extend the monitoring capabilities of the system by enabling you to create user-defined attributes for monitoring.

Extending the system in this way involves creating a microsensor.

Table 38 describes the classes defined by the microsensor resource manager.

Table 38. Resource classes defined by the microsensor resource manager

Resource class	Description
IBM.Microsensor	Each resource of this class represents a microsensor, a dynamically-loaded shared library that provides C-based functions, which the RMC runs to: • Obtain control information about the microsensor. • Start the monitoring of some user-defined values. • Stop the monitoring of some user-defined values. • Retrieve the user-defined values. • Notify the library that it is no longer needed.

The microsensor is essentially a resource that you add to the microsensor resource class of the microsensor resource manager. The values returned by the functions are dynamic attributes of that resource. Using the event response resource manager commands, you can then create a condition to monitor any of the attributes you have defined. The microsensor resource manager provides a command-line interface for creating, changing, listing, and removing microsensors.

Complete syntax information on the commands is provided in *RSCT for AIX: Technical Reference* and *RSCT for Multiplatforms: Technical Reference*.

For information on listing attribute values or attribute definitions for the resource class (or resource instances of the resource classes) shown in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the sensor resource manager:

In general, you will not need to manipulate resources of this class directly. Instead, you will use the sensor resource manager commands.

Table 39 describes the resource class defined by the sensor resource manager. In general, you will not need to manipulate resources of this class directly. Instead, you will use the sensor resource manager commands described in "Creating event sensor commands for monitoring" on page 177.

Table 39. Resource classes defined by the sensor resource manager

Resource class	Description
IBM.Sensor	Each resource of this class represents a sensor (a command that the RMC runs to retrieve one or more user-defined values).

For information on listing attribute values or attribute definitions for the resource class (or resource instances of the resource classes) shown in the preceding table, see "Listing resource information" on page 130.

Resource classes defined by the storage resource manager:

The resource classes represent physical disks and related storage entities. The storage resource manager uses these representations to protect the data integrity of critical disk resources (resources shared across two or more nodes) in an RSCT peer domain.

Table 40 describes the resource classes defined by the storage resource manager.

Table 40. Resource classes defined by the storage resource manager

Resource class	Description
IBM.AgFileSystem	This resource class externalizes the attributes of any file systems on a Linux disk partition resource (IBM.Partition), an entire Linux disk resource (IBM.Disk), or a Linux or AIX logical volume resource (IBM.LogicalVolume). These attributes are a subset of the entries in the IBM.FileSystem class of the File System resource manager. **Note:** Supported on Solaris.
IBM.Disk	This resource class externalizes the attributes of SCSI disks and MD devices on Linux and physical volumes that are members of a volume group.
IBM.LogicalVolume	This resource class externalizes the attributes of logical volumes configured in volume groups.
IBM.Partition	On Linux nodes only, this resource class externalizes the attributes of any configured partitions on a disk device resource of the IBM.Disk class.
IBM.VolumeGroup	This resource class externalizes the attributes of volume groups comprised of one or more physical volumes.

Note: Only the IBM.AgFileSystem class is supported on Solaris.

For information on listing attribute values or attribute definitions for the resource class (or resource instances of the resource classes) shown in the preceding table, see "Listing resource information" on page 130. For more information on the Storage resource manager, see "Administering the storage resource manager" on page 239.

How RMC and the resource managers enable you to manage resources

RMC provides resource class abstractions for representing physical or logical system entities, while the individual resource managers map actual entities to these abstractions.

Since the various resource managers all define resources according to the same abstractions defined by RMC, RMC is able to manage the resources generically. RMC provides a set of commands that enable you to list information about and manipulate resources, regardless of which resource manager defines the particular resource class.

Often these general RMC commands are not needed. For example, a **mkrsrc** command exists, enabling you to define a new resource of a particular class. However, the resource managers often automatically harvest this information to create the resources, or certain resource manager commands explicitly or implicitly create the resource. For example, the event response resource manager provides the **mkcondition** command to create a condition for resource monitoring. The **mkcondition** command creates a Condition resource; there is no need to use the generic **mkrsrc** command.

The RMC commands you will use most commonly are the **lsrsrc** and **lsrsrcdef** commands which display resource or resource class information you may need when issuing other commands. The **lsrsrc** command lists the persistent and/or dynamic attributes of resources, and the **lsrsrcdef** lists a resource class definition. For more information on the **lsrsrc** and **lsrsrcdef** commands, see "Listing resource information" on page 130.

How RMC and the resource managers enable you to monitor resources

RMC and the resource managers together provide sophisticated monitoring and response capabilities that enable you to detect, and in many cases correct, system resource problems such as a critical file system becoming full.

You are able to monitor virtually all aspects of your system resources and specify a wide range of actions to take — from general notification or logging capabilities we provide to more targeted recovery responses you define.

The resource monitoring capability is largely provided by the event response resource manager (although you are typically monitoring dynamic attribute values provided by the host resource manager, file system resource manager, and sensor resource manager). The event response resource manager provides a set of commands that enable you to monitor events of interest (called *conditions*) and have the RMC system react in particular ways (called *responses*) if the event occurs.

A condition

A *condition* specifies the event that should trigger a response. It does this using an *event expression* and, optionally, a *rearm event expression*.

One or more attributes can be used in an expression. Attributes can be a combination of dynamic and persistent attributes. Some examples follow.

The following expression generates an event notification if, during any sampling interval, the number of page-ins exceeds twice the number of page-outs. Note that the **P** suffix indicates the previous value. The RMC subsystem maintains the *N*th and *N*th -1 value of each monitored attribute.

```
(VMPgInRate@R - VMPgInRate@RP) > 2 * (VMPgOutRate@R - VMPgOutRate@RP)
```

The following expression generates an event when the file system is almost full, but the number of files in the file system is relatively small.

```
PercentTotUsed > 95 && PercentINodeUsed < 5
```

The following example uses **PercentTotUsed** in the primary expression and **Size** in the re-arm expression.

```
PercentTotUsed > 90
```

```
Size > Size@P && Size@P != 0
```

In this case, a "file system full" event is generated when the file system is greater than 90%, but the RMC subsystem does not start looking for that condition again until the file system has been made larger. The check for the previous value of **Size** is to handle the start monitoring case, where there is no previous value.

Event expressions:

An *event expression* consists of an attribute name, a mathematical comparison symbol, and a constant.

For example, the IBM.FileSystem resource class defines a dynamic attribute PercentTotUsed to represent the percentage of space used in a file system. The following event expression, if specified on a condition, would trigger an event if a file system resource in the resource class was over 90 percent full:

```
PercentTotUsed > 90
```

The condition's event expression will, by default, apply to all resources of a particular resource class (in this example, all file systems). However, using a selection string that filters the resources based on persistent attribute values, you can create a condition that applies only to a single resource of the resource class or a subset of its resources. For example, the following selection string, if specified on a condition, would specify that the condition applies only to the **/var** file system. This selection string uses the persistent attribute Name of the resource class to identify the **/var** file system.

```
"Name == \"/var\""
```

Our condition now will now trigger an event only if the **/var** file system is over 90 percent full. When the condition is later active, RMC will periodically test the event expression at set intervals to see if it is true. If the expression does test true, RMC triggers any responses associated with the condition.

As already stated, each event expression refers to a particular attribute value (usually a dynamic attribute), which will be polled by RMC at set intervals to determine if the expression tests true. RMC keeps track of the previously observed value of the attribute, so the event expression can compare the currently observed value with the previously observed value. If the event expression suffixes the attribute name with "@P", this represents the previously observed value of the attribute. For example, the following event expression, if specified on a condition, would trigger an event if the average number of processes on the run queue has increase by 50% or more between observations.

```
(ProcRunQueue - ProcRunQueue@P) >= (ProcRunQueue@P * 0.5)
```

In attribute value comparison, some operators may not be applicable depending on the variable type. For example, the following event expression, if specified on a condition, would trigger an event when a program generates a set of unique random integers:

```
Int32 != Int32@P
```

In this example, the != , < , and > operators are all valid because Int32 is a *state* variable. The value of a state variable must change in order for an event to be triggered. If the event expression `Int32 == Int32@P` is used, the event will fail to trigger because the expression indicates that the previously observed value is the same as the current value.

To identify the variable type of an attribute, use the **lsrsrcdef** command. For example, to find the variable type of Int32 for an IBM.Sensor resource, you would enter the following at the command line:

```
lsrsrcdef -Ad IBM.Sensor Int32
```

Rearm event expressions:

A condition can optionally have a *rearm event expression* defined. If it does, then RMC will stop evaluating the event expression once it tests true, and instead will evaluate the rearm event expression until it tests true. Once the rearm event expression tests true, the condition is rearmed.

In other words, RMC will once again evaluate its event expression. For example, our event expression tests to see if the **/var** file system is 90 percent full. If it is, the associated response is triggered. We might not want RMC to continue evaluating this same expression and so triggering the same response over and over. If the response was to notify you by e-mail of the condition, the first e-mail would be enough. That's where a rearm event expression comes in. The following expression, if specified as the condition's rearm event expression, will rearm the condition once the **/var** file system in less than 75 percent full.

```
PercentTotUsed < 75
```

Figure 1 illustrates the cycle of event expression/rearm event expression evaluation.

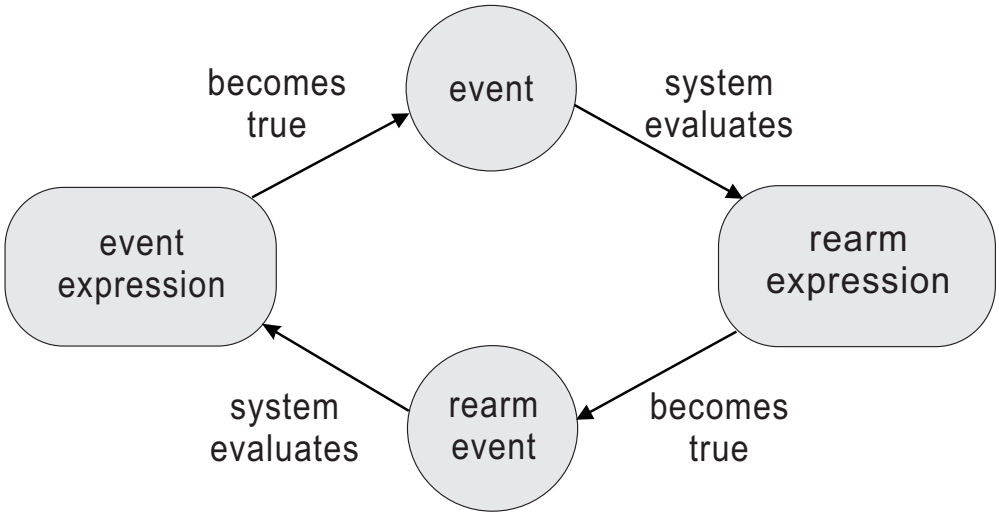

Figure 1. Cycle of event expression and rearm event expression evaluation for a condition

The monitoring scope of a condition:

Another important feature of a condition, the *monitoring scope* refers to the node or set of nodes where the condition is monitored.

Although a condition resource is defined on a single node, its monitoring scope could be the local node only, all the nodes of a peer domain, select nodes of a peer domain, all the nodes of the management domain, or select nodes of a management domain. If the monitoring scope indicates nodes of a peer domain, the node on which the condition resource is defined must be part of the peer domain. If the monitoring scope indicates nodes of a management domain, the node on which the condition resource is defined must be the management server of the management domain.

Predefined and user-defined conditions:

It is important to understand that, in most cases, you will not need to create conditions. This is because RSCT provides a set of predefined conditions to monitor most of the dynamic attributes defined by the file system resource manager and host resource manager.

You can list these predefined conditions using the **lscondition** command described in "Listing conditions" on page 137.

If the predefined conditions are not sufficient, you can create your own conditions to monitor any attribute. To do this, you use the **mkcondition** command as described in "Creating a condition" on page 157. Even if you are creating your own conditions, you can usually copy one of the predefined ones to use as a template, modifying it as you see fit. If none of the existing attributes contains the value you are interested in monitoring, you can extend the RMC system by creating a sensor. A *sensor* is merely a command that the RMC system runs (at specified intervals and/or when you explicitly request for it to be run) to retrieve one or more user-defined values. For more information, see "Creating event sensor commands for monitoring" on page 177.

A response

A *response* indicates one or more actions that the system can take when a condition event occurs.

A *condition event* occurs when a condition's event expression or rearm event expression tests true. When such an event occurs, a response associated with the condition is triggered and any number of its *response actions* can execute.

A *response action* is a command or script that responds to the condition event. These response actions could perform a general-purpose action such sending e-mail notifying you of the event, or logging the event information to a file. In fact we provide several predefined action scripts that perform such general-purpose actions. You can also write your own scripts to provide more specific responses to events. For example, if a condition tests to see if a directory is over 90 percent full, an associated response action could automatically delete the oldest unnecessary files in the directory.

A response can have multiple actions, enabling the system to respond one way to a condition event and another way to a condition rearm event. Figure 2 illustrates a response with multiple actions, one for the condition event and another for the condition rearm event.

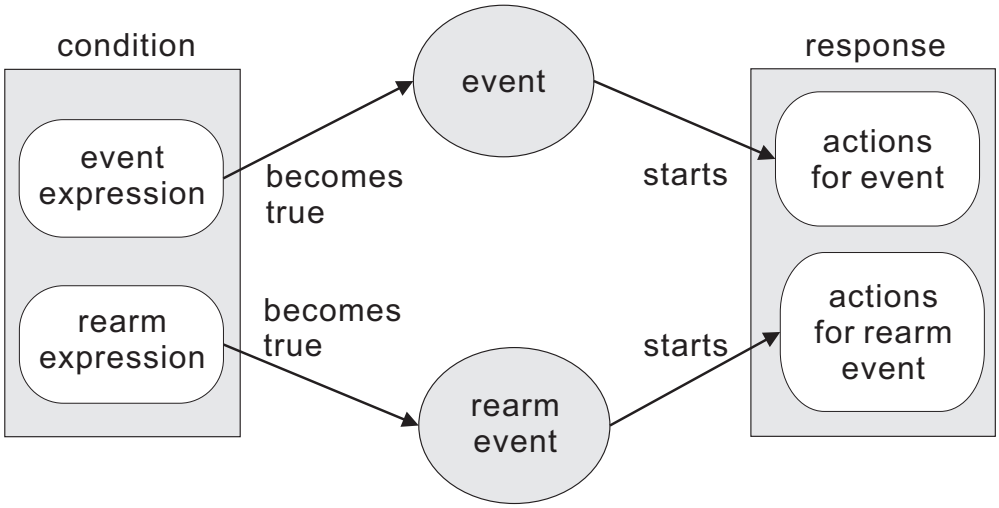

Figure 2. A response with separate actions for a condition's event and rearm event

Having multiple actions also enables a response to behave differently based on the day of the week and time of day that the event occurs. One response action might be triggered on weekdays during working hours, while another might be triggered on the weekends and on weekdays outside working hours. For example, suppose that you have a condition that will trigger an event if a processor goes offline. During working hours, you might want the system to send you e-mail when this happens. Outside work hours, the system could instead log the information to a file that you check when you come into the office.

Predefined and user-defined responses:

Think of a response as a container for one or more actions that the system can take when an associated condition event occurs.

Using the **mkresponse** command (as described in "Creating a response" on page 168), you can add a single action to the response. You can then use the **chresponse** command (as described in "Modifying a response" on page 175) to add more actions to the response.

Just as RSCT provides a set of predefined conditions that you can use, it also provides a set of predefined responses. These responses utilize predefined action scripts that RSCT also provides. Table 41 describes the predefined responses that RSCT provides.

Table 41. Predefined responses that RSCT provides

Response name	Response action	Description	Response action in effect
Broadcast event on-shift	Broadcast message	Uses the predefined action script **/usr/sbin/rsct/bin/wallevent** to broadcast an event or rearm event to all users that log in to the host.	8 a.m. - 5 p.m., Monday to Friday
Broadcast details of event any time	Broadcast event details	Available on Linux nodes only. Uses the predefined action script **/usr/sbin/rsct/bin/wallevent** to broadcast an event or rearm event to all users that log in to the host. Specifies the **wallevent** script's **-c** flag to broadcast event details.	All day, everyday
Email root off-shift	Email root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an email to root when an event or a rearm event occurs.	5 p.m. - 12 p.m., Monday to Friday 12 a.m. - 8 a.m., Monday to Friday All day, Saturday and Sunday
Email root anytime	Email root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an email to root when an event or a rearm event occurs.	All day, everyday
Log event anytime	Log event	Uses the predefined action script **/usr/sbin/rsct/bin/logevent** to log an entry to **/tmp/systemEvents** whenever an event or a rearm event occurs.	All day, everyday
Informational notifications	Log info event	Uses the predefined action script **/usr/sbin/rsct/bin/logevent** to log an entry to **/tmp/infoEvents** whenever an event or a rearm event occurs.	All day, everyday
	Email root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an email to root when an event or a rearm event occurs.	8 a.m. - 5 p.m., Monday to Friday
Warning notifications	Log warning event	Uses the predefined action script **/usr/sbin/rsct/bin/logevent** to log an entry to **/tmp/warningEvents** whenever an event or a rearm event occurs.	All day, everyday
	Email root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an email to root when an event or a rearm event occurs.	All day, everyday
Critical notifications	Log critical event	Uses the predefined action script **/usr/sbin/rsct/bin/logevent** to log an entry to **/tmp/criticalEvents** whenever an event or a rearm event occurs.	All day, everyday
	Email root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an email to root when an event or a rearm event occurs.	All day, everyday

Table 41. Predefined responses that RSCT provides (continued)

Response name	Response action	Description	Response action in effect
	Broadcast message	Uses the predefined action script **/usr/sbin/rsct/bin/wallevent** to broadcast an event or rearm event to all users that log in to the host.	All day, everyday
Generate SNMP trap	SNMP trap	Uses the predefined action script **/usr/sbin/rsct/bin/snmpevent** to send a Simple Network Management Protocol (SNMP) trap of an ERRM event to a host running an SNMP agent.	All day, everyday

A condition-response association

Before you can actually monitor a condition, you must link it with one or more responses. This is called a *condition-response association* and is required for monitoring so that RMC knows how to respond when the condition event occurs.

You can create a condition/response association using either the **mkcondresp** or **startcondresp** commands. The **mkcondresp** command makes the association, but does not start monitoring it. The **startcondresp** command either starts monitoring an existing association, or defines the association and starts monitoring it.

What to monitor

To get an idea of what you can monitor, look at the predefined conditions that RSCT provides. You can list the predefined conditions using the **lscondition** command.

The **lscondition** command is described in "Listing conditions" on page 137.

You can also create a condition based on any attribute of a resource class. Because persistent attributes do not change, in general, it makes the most sense to monitor a dynamic attribute. You can list the dynamic attributes using the **lsrsrc** command (described in "Displaying attribute value information for a resource or a resource class" on page 133) and the **lsrsrcdef** command (described in "Displaying attribute definition information for a resource or a resource class" on page 135).

Keep in mind that certain cluster licensed programs, such as Tivoli System Automation, provide additional resource managers. These resource managers might have resource classes with their own predefined conditions and their own attributes. See the documentation for these licensed programs for information about any predefined conditions or attributes they provide.

IBM recommends that you monitor the size of the **/var** file system because many RSCT subsystems use this file system extensively. To monitor the **/var** file system, you can use the predefined condition **/var space used**, which is provided by the file system resource manager.

How RMC implements authorization

RMC implements authorization using an Access Control List (ACL) file. Specifically, RMC uses the ACL file on a particular node to determine the permissions that a user must have in order to access particular resource classes and their resource instances on that node.

For example, in order to modify a persistent attribute for an instance of a resource class on a particular node, the user must have write permission for that resource class on that node. To monitor an attribute, the user must have read permission. A node's RMC ACL file is named **ctrmc.acls** and is installed in the directory **/usr/sbin/rsct/cfg**. You can have RMC use the default permissions set in this file, or you can modify it after copying it to the directory **/var/ct/cfg** as described in "Managing user access to resources using RMC ACL files" on page 126.

Target nodes for a command

Resource monitoring and control (RMC) is a daemon that runs on individual systems or on each node of a cluster. It provides a single management and monitoring infrastructure for individual nodes, peer domains, and management domains.

You can execute RMC and resource manager commands on a single node, on all the nodes of a peer domain, or on all the nodes of a management domain. Some commands refine this selection of nodes even further such that you can specify a subset of nodes in the peer domain or management domain. When you work in a cluster, you can also issue commands from a local node to be executed on another node.

Note: If RSCT 3.1.5.0, or later, is installed on your operating system, the RMC subsystem is not recycled on nodes when the node is brought online or taken offline from a peer domain. If the local or management scope is specified, RMC and the execution of resource manager commands are not interrupted by the peer domain transition.

The following environment variables, along with various command flags, determine the nodes that are affected by the RMC and resource manager commands that you enter.

Table 42. Environment variables that determine target nodes for a command

Environment variable	Description
CT_CONTACT	Determines the system where the session with the RMC daemon occurs. When set to a host name or IP address, the command contacts the RMC daemon on the specified host. If not set, the command contacts the RMC daemon on the local system where the command is being run.
CT_MANAGEMENT_SCOPE	Identifies the management scope. The management scope determines the set of possible target nodes for the command. The default is local scope. The valid values follow: **0** The local scope. It is either the local node or the node indicated by the CT_CONTACT environment variable. **1** The local scope. It is either the local node or the node indicated by the CT_CONTACT environment variable. **2** The peer domain scope. It is either the peer domain in which the local node is online, or the peer domain in which the node indicated by the CT_CONTACT environment variable is online. **3** The management domain scope.

Not all of the RMC and resource manager commands use these environment variables, but the ones that do might have command-line flags that you can use to override the environment variable setting or otherwise determine how the command uses the specified values.

When the scope is set to management domain scope (either through the CT_MANAGEMENT_SCOPE environment variable or through command-line options), RMC commands that are issued from the management server return information for managed nodes. Some of these nodes might also be in peer domains within the management domain.

Certain RMC class operations return information about a node's peer domain. When you perform these operations in a management domain scope, some nodes might not be in a peer domain. In these cases, the peer domain field provides only the local host name. When a local host name is provided instead of a peer domain name, the name appears in angle brackets (for example, <*local_host_name*>).

The *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides contain complete reference information for all of the RSCT commands. The reference information contains details on how each command uses these environment variables. The same reference information can be found for any command by viewing its online man page.

Targeting nodes:

When this documentation refers a command, it focuses on the command's basic functions (such as listing a condition, starting monitoring, viewing an audit log) and does not cover targeting nodes in the body of the discussion. However, many commands can target the local node, a remote node, a group of nodes in a peer domain, an entire peer domain, a node in a management domain, and so on. Where appropriate, any information on how the particular command handles the targeting of nodes is covered in a separate **Targeting Nodes** note.

Related concepts:

"Management domains and peer domains" on page 3
The set of nodes that is configured for manageability or monitoring is called a *management domain* of your cluster. The set of nodes that is configured for high availability is called an RSCT peer domain of your cluster.

Monitoring resources using RMC and resource managers

The Resource Monitoring and Control (RMC) subsystem is the scalable backbone of RSCT that provides a generalized framework for managing and monitoring resources (physical or logical system entities) within a single system or a cluster.

Note: Most of the predefined conditions described in the documentation are not available in the Linux implementation of RSCT. However, these same conditions can easily be created by following the instructions in "Creating a condition" on page 157.

RMC is a daemon that runs on individual systems or each node of a cluster. It provides a single management/monitoring infrastructure for individual machines, peer domains, and management domains. RMC, however, is a generalized framework — it provides an abstract way of representing resources of a system, but it does not itself represent the actual resources. The actual resources are represented by resource managers. A resource manager is a daemon process that maps RMC's resource abstractions into actual descriptions of resources. Since the various resource managers all define resources according to the same abstraction defined by RMC, RMC is able to manage the resources generically.

Related concepts:

"An RSCT peer domain" on page 35
An *RSCT peer domain* is a cluster of nodes configured for high availability.

Managing user access to resources using RMC ACL files

RMC implements authorization using an *access control list* (ACL) file. Specifically, RMC uses the ACL file on a particular node to determine the permissions that a user must have in order to access resource classes and their resource instances.

A node's RMC ACL file is named **ctrmc.acls** and is installed in the directory **/usr/sbin/rsct/cfg**. You can allow RMC to use the default permissions set in this file or you can modify the file after copying it to the directory **/var/ct/cfg/**, as described in "Modifying the RMC ACL file" on page 129.

Note: You cannot use an RMC ACL file to determine the permissions for the IBM.LPCommands resource class. The IBM.LPCommands resource class is implemented by the least-privilege resource manager and is used to represent a root command or script that only authorized users can run. Because the least-privilege resource manager is designed to grant user's authority to run only certain root commands without giving them full root authority, the least-privilege resource manager has its own set of access control lists to provide a finer level of authorization for individual IBM.LPCommands resources.

For more information, see "Controlling access to root commands and scripts" on page 214.

Related concepts:

"Overview of the LP resource manager's access control lists" on page 215
Although authorization for most resource classes and resources is determined by settings in an RMC ACL file, this is not true for the IBM.LPCommands resource class or resources of the **IBM.LPCommands** resource class.

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

Format of an RMC ACL file

An RMC ACL file has a stanza format that consists of a stanza name followed by 0 or more stanza lines.

The RMC ACL file appears as follows:

```
stanza_name
     user_identifier     type     permissions
     user_identifier     type     permissions
⋮
     user_identifier     type     permissions
```

A stanza begins with a line containing the stanza name, which is the name of a resource class, the keyword OTHER, or the keyword DEFAULT. The stanza name must start in column one. A stanza is terminated by a blank line, a comment line, another stanza, or the end-of-file.

The OTHER stanza applies to all resource classes that are not otherwise specified in the file. The lines in the DEFAULT stanza are implicitly appended to the stanzas for each resource class specified in the ACL file, including the OTHER stanza. If the OTHER stanza is not specified, then the permissions of any resource class not specified in this file will be the permissions specified in the DEFAULT stanza.

Following the line containing the stanza name, the remaining lines of the stanza, if any, consist of leading white space (one or more blanks, tabs, or both) followed by one or more white space separated tokens that include:

- A user identifier
- An object type
- An optional set of permissions

A comment line begins, after optional white space, with the **#** character or the **//** characters. Stanza lines may have trailing comments that are specified by these characters. The trailing comment must be preceded by white space, as follows:

```
stanza_name                                      // trailing comment
     user_identifier     type     permissions     // trailing comment
     user_identifier     type                     // no permissions
```

The `user_identifier` portion of the stanza line can have any one of these forms.

Table 43. The user identifier portion of the stanza line

This form of user identifier...	Specifies...
host:*host_user_identifier*	A host user identifier. The **host:** keyword is optional. It specifies that the user identifier can be matched against a network identifier provided by the host based authentication (HBA) security mechanism (described in "Configuring cluster security services" on page 267). If the **host:** keyword is omitted and the entry does not take one of the other forms outlined in this table, the entry is assumed to be a host user identifier. The host user identifier can take a number of different formats: *user_name*@*host_identifier* Specifies a particular user. The *host_identifier* portion of this specification can take a number of forms. These forms are the same as when the host user identifier format is specified as a *host_identifier* alone, and are described below. *host_identifier* Specifies any user running the RMC application on the host identified. The *host_identifier* can be: • A fully qualified host name • A short host name • An IP address • A node ID. This is a 16-digit hexadecimal number. For example, 0xaf58d41372c47686. • The keyword LOCALHOST. This keyword identifies the local host. * Specifies any user running an RMC application on any host.
none:*mapped_user_identifier*	A mapped name as specified in the **ctsec_map.global** or **ctsec_map.local** file. See "Configuring the host based authentication (HBA) mechanism mappings" on page 281 for more information on creating these mapped names.
UNAUTHENT	An unauthenticated user.

The stanza, including lines implicitly appended from the DEFAULT stanza, is examined in two passes. The first pass attempts to match a line against the user's network ID. If no match can be made, then a second pass is performed in an attempt to match a line against the user's mapped ID.

The next part of the stanza is the type; it can be any of the characters shown in Table 44.

Table 44. The type portion of the stanza line

Specifying this type...	Indicates that the permissions provide access to...
C	The resource class
R	All resource instances of the class
*	Both the resource class and all instances of the class

The final part of the stanza line is the optional permissions, described in Table 45 on page 129.

Table 45. The optional permissions portion of the stanza line

Specifying this permission...	Indicates that the specified users at the specified hosts have...
r	Read permission. This allows the users to register and unregister events, query attribute values, and validate resource handles. The **r** permission is a composite permission that is composed of the following permissions. While you could, instead of specifying the **r** permission, specify a subset of the following permissions, this would prevent the user from performing some operations. The **r** permission is a convenient way of specifying all of the following: **q** Indicates that the specified user at the specified host has query permission. This allows the user to query persistent or dynamic attributes. **l** Indicates that the specified user at the specified host has list permission. This allows the user to list resources. **e** Indicates that the specified user at the specified host has event permission. This allows the user to register, query, and unregister events. **v** Indicates that the specified user at the specified host has validate permission. This allows the user to validate resource handles.
w	Write permission. This allows the user(s) to run all other command interfaces. The **w** permission is a composite permission that is composed of the following permissions. While you could, instead of specifying the **w** permission, specify a subset of the following permissions, this would prevent the user from performing some operations. The **w** permission is a convenient way to specify all of the following: **d** Indicates that the specified user at the specified host has define permission. This allows the user to define and undefine resources. **c** Indicates that the specified user at the specified host has refresh permission. This allows the user to refresh resource configuration. **s** Indicates that the specified user at the specified host has set permission. This allows the user to set attributes. **o** Indicates that the specified user at the specified host has online permission. This allows the user to bring resources online and take resources offline.
rw	Read and write permission.

If the permissions are omitted, then the user does not have access to the objects specified by the *type* character. Note that no permissions are needed to query resource class and attribute definitions.

For any command issued against a resource class or its instances, the RMC subsystem examines the lines of the stanza matching the order specified in the ACL file. The first line that contains an identifier that matches the user issuing the command and an object type that matches the objects specified by the command is the line used in determining access permissions. Therefore, lines containing more specific user identifiers and object types should be placed before lines containing less specific user identifiers and object types.

Modifying the RMC ACL file

When RMC is installed on a node, a default ACL file is provided in **/usr/sbin/rsct/cfg/ctrmc.acls**. *This file should not be modified.*

It contains the following default permissions:

```
IBM.HostPublic
      *             *      r
      UNAUTHENT     *      r

DEFAULT
   root@LOCALHOST * rw
   LOCALHOST * r
```

The first stanza enables anyone to read the information in the IBM.HostPublic class, which provides information about the node, mainly its public key. The second stanza contains default ACL entries. It grants, for this node, read/write permission to root and read-only permission to any other user.

To change these defaults:

1. Copy the file **/usr/sbin/rsct/cfg/ctrmc.acls** to **/var/ct/cfg/ctrmc.acls**.
2. Use the **chrmcacl** command to update the **/var/ct/cfg/ctrmc.acls** file.
3. Activate your new permissions using the following **refresh** command:

   ```
   refresh -s ctrmc
   ```

   Provided there are no errors in the modified ACL file, the new permissions will take effect. If errors are found in the modified ACL file, then the contents of the file are ignored and the previously-defined permissions remain in effect. The ACL file errors are logged to **/var/ct/IW/log/mc/default**.

Listing resource information

The **lsrsrc** and **lsrsrcdef** commands enable you to list information about the resources available on your system or cluster.

Specifically, you can:

- Issue either the **lsrsrc** or **lsrsrcdef** command without any command parameters or flags to obtain a list of resource classes available on your system or cluster.
- Use the **lsrsrc** command to list the values of resource or resource class attributes.
- Use the **lsrsrcdef** command to list attribute definitions for a resource or resource class.

Related concepts:

"Resource classes defined by the configuration resource manager" on page 114
In general, you do not need to manipulate resources of these classes directly. Instead, use the configuration resource manager commands.

Listing available resource classes

To display a list of the resource classes on your cluster or system, issue either the **lsrsrc** or **lsrsrcdef** command without any command parameters or flags.

For example, enter the following command:

```
lsrsrc
```

The following output is displayed:

```
class_name
"IBM.AgFileSystem"
"IBM.Association"
"IBM.ATMDevice"
"IBM.AuditLog"
"IBM.AuditLogTemplate"
"IBM.CommunicationGroup"
"IBM.Condition"
"IBM.Disk"
"IBM.EthernetDevice"
"IBM.EventResponse"
"IBM.FDDIDevice"
"IBM.Host"
"IBM.HostPublic"
"IBM.FileSystem"
"IBM.HeartbeatInterface"
"IBM.LogicalVolume"
"IBM.LPCommands"
"IBM.MCP"
```

```
"IBM.MicroSensor"
"IBM.MngNode"
"IBM.MngNodeNetIF"
"IBM.NetworkInterface"
"IBM.PagingDevice"
"IBM.Partition"
"IBM.PeerDomain"
"IBM.PeerNode"
"IBM.PhysicalVolume"
"IBM.Processor"
"IBM.Program"
"IBM.PublicKeyExchange"
"IBM.RMCCtrl"
"IBM.RSCTParameters"
"IBM.Sensor"
"IBM.TieBreaker"
"IBM.TokenRingDevice"
"IBM.VolumeGroup"
```

To return detailed information on any of the resource classes, see the instructions in "Displaying attribute value information for a resource or a resource class" on page 133 and "Displaying attribute definition information for a resource or a resource class" on page 135.

Table 46 lists the resource classes defined by the RSCT resource managers.

Table 46. Resource classes provided by RSCT resource managers

This resource class...	Is defined by this resource manager...	For more information, see...
IBM.AgFileSystem	Storage resource manager	"Resource classes defined by the storage resource manager" on page 118
IBM.Association	Event response resource manager	"Resource classes defined by the event response resource manager" on page 115
IBM.ATMDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.AuditLog	Audit log resource manager	"Resource classes defined by the audit log resource manager" on page 113
IBM.AuditLogTemplate	Audit log resource manager	"Resource classes defined by the audit log resource manager" on page 113
IBM.CommunicationGroup	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.Condition	Event response resource manager	"Resource classes defined by the event response resource manager" on page 115
IBM.Disk	Storage resource manager	"Resource classes defined by the storage resource manager" on page 118
IBM.EthernetDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.EventResponse	Event response resource manager	"Resource classes defined by the event response resource manager" on page 115
IBM.FDDIDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.FileSystem	File system resource manager	"Resource classes defined by the file system resource manager" on page 115\
IBM.HeartbeatInterface	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.Host	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.HostPublic	Host resource manager	"Resource classes defined by the host resource manager" on page 115

Table 46. Resource classes provided by RSCT resource managers (continued)

This resource class...	Is defined by this resource manager...	For more information, see...
IBM.LogicalVolume	Storage resource manager	"Resource classes defined by the storage resource manager" on page 118
IBM.LPCommands	Least-privilege resource manager	"Resource classes defined by the least-privilege resource manager" on page 116
IBM.MCP	Management domain resource manager	"Resource classes defined by the management domain resource manager" on page 116
IBM.MicroSensor	Microsensor resource manager	"Resource classes defined by the microsensor resource manager" on page 117
IBM.MngNode	Management domain resource manager	"Resource classes defined by the management domain resource manager" on page 116
IBM.MngNodeNetIF	Management domain resource manager	"Resource classes defined by the management domain resource manager" on page 116
IBM.NetworkInterface	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.PagingDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.Partition	Storage resource manager	"Resource classes defined by the storage resource manager" on page 118
IBM.PeerDomain	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.PeerNode	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.PhysicalVolume	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.Processor	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.Program	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.PublicKeyExchange	Management domain resource manager	"Resource classes defined by the management domain resource manager" on page 116
IBM.RMCCtrl	Management domain resource manager	"Resource classes defined by the management domain resource manager" on page 116
IBM.RSCTParameters	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.Sensor	Sensor resource manager	"Resource classes defined by the sensor resource manager" on page 118
IBM.TieBreaker	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 114
IBM.TokenRingDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 115
IBM.VolumeGroup	Storage resource manager	"Resource classes defined by the storage resource manager" on page 118

In addition to the resource classes listed in Table 46 on page 131, keep in mind that:

- Any resource classes that are registered using the CIM resource manager (described in "Querying and monitoring CIM properties and associations" on page 184) might also appear in the preceding **lsrsrc** command output. When a CIM class is registered, it is mapped to a new RMC resource class. The RMC resource class name is a concatenation of the namespace and the CIM class name, for example: **cimv2.Linux_ComputerSystem**. All registered CIM classes are placed in the **root/cimv2** namespace.
- Additional resource managers are provided by licensed programs that exploit the RSCT technology, such as Tivoli System Automation. If you have any RSCT exploiter programs installed on your system or cluster, the preceding command output might include additional resource classes that are defined by resource managers of the RSCT exploiters. For information about any additional resource classes that are not listed in the preceding table, see the appropriate RSCT exploiter documentation.

For more information about the **lsrsrc** and **lsrsrcdef** commands, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Displaying attribute value information for a resource or a resource class

You can display attribute value information for a resource or a resource class by issuing the **lsrsrc** command with the name of a resource class.

Various command options, as shown in Table 47, enable you to display attribute information for:

- A resource class or resource instances of the class
- Persistent attributes, dynamic attributes, or both persistent and dynamic attributes

Table 47. Displaying attribute value information for a resource or resource class

To display attribute value information for...	Persistent attributes	Dynamic attributes	Both persistent and dynamic attributes:
A resource class:	**lsrsrc -c -A p** *Resource_Class*	**lsrsrc -c -A d** *Resource_Class*	**lsrsrc -c -A b** *Resource_Class*
A resource:	**lsrsrc -A p** *Resource_Class*	**lsrsrc -A d** *Resource_Class*	**lsrsrc -A b** *Resource_Class*

The **lsrsrc** command returns a list of the attributes requested. The attribute name and value are listed.

Example: To list the persistent and dynamic attributes for the resources of the **IBM.Host** class, enter:

```
lsrsrc -A b IBM.Host
```

The following output is displayed:

```
Resource Persistent and Dynamic Attributes for IBM.Host
resource 1:
        Name               = "jbrady.ibm.com"
        NumProcessors      = 4
        RealMemSize        = 1073676288
        OSName             = "AIX"
        KernelVersion      = "7.1"
        DistributionName   = "IBM"
        DistributionVersion = "7100-00-00"
        Architecture       = "ppc"
        NumOnlineProcessors = 4
        ActivePeerDomain   = "JoeMixed"
        NodeNameList       = {"jbrady.ibm.com"}
        ProcRunQueue       = 1.01167
        ProcSwapQueue      = 1.01822
        TotalPgSpSize      = 131072
        TotalPgSpFree      = 130845
        PctTotalPgSpUsed   = 0.173187
        PctTotalPgSpFree   = 99.8268
        PctTotalTimeIdle   = 95.0711
        PctTotalTimeWait   = 0.152439
        PctTotalTimeUser   = 4.06504
        PctTotalTimeKernel = 0.711382
```

```
        PctRealMemFree      = 58
        PctRealMemPinned    = 8
        RealMemFramesFree   = 153110
⋮
```

Example: To list the persistent attributes for the resources of the **IBM.PeerDomain** class, enter:

```
lsrsrc IBM.PeerDomain
```

The following output is displayed:

In a CAA environment:

```
Resource Persistent Attributes for IBM.PeerDomain
resource 1:
        Name                = "my_caa"
        RSCTActiveVersion   = "3.1.5.0"
        RealMemSize         = 1073676288
        MixedVersions       = 0
        TSPort           = 12347
        GSPort              = 12348
        RMCPort             = 657
        ResourceClasses     = {}
        QuorumType          = 4
        DomainType          = 1
        ActivePeerDomain    = "my_caa"
```

In a non-CAA environment:

```
Resource Persistent Attributes for IBM.PeerDomain
resource 1:
        Name                = "RPD1"
        RSCTActiveVersion   = "3.1.5.0"
        MixedVersions       = 0
        TSPort              = 12347
        GSPort              = 12348
        RMCPort             = 657
        ResourceClasses     = {}
        QuorumType          = 0
        DomainType          = 0
        ActivePeerDomain    = "RPD1"
```

Note: A `DomainType` attribute of 1 indicates that RPD is created in CAA environment by using the **mkrpdomain** command with the **–C** option. A `DomainType` attribute of 0 indicates that the RPD is created by using the **mkrpdomain** command without the **–C** option.

Example: To list the definition of the `DomainType` resource persistent attribute of the **IBM.PeerDomain** class, enter:

```
lsrsrcdef -e IBM.PeerDomain DomainType
```

The following output is displayed:

```
Resource Persistent Attribute Definitions for IBM.PeerDomain
attribute 1:
        program_name      = "DomainType"
        display_name      = "Domain Type"
        group_name        = "Basic (Group 0)"
        properties        = {"read_only","option_for_define","selectable","public"}
        description       = "Domain Type"
        attribute_id      = 10
        group_id          = 0
        data_type         = "uint32"
        variety_list      = {[1,1]}
        variety_count     = 1
        default_value     = 0
```

Although the attribute names are often self-explanatory, you can use the **lsrsrcdef** command to display definition information (including a description) for the attributes listed. The **lsrsrcdef** command is described in "Displaying attribute definition information for a resource or a resource class."

Targeting nodes:

The **lsrsrc** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon carries out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **lsrsrc** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope. If the CT_MANAGEMENT_SCOPE environment variable is not set and the **-a** flag is specified, then the default management scope is the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lsrsrc** command man page and "Target nodes for a command" on page 125.

For complete syntax information about the **lsrsrc** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* or *Technical Reference: RSCT for Multiplatforms*.

Displaying attribute definition information for a resource or a resource class

You can display attribute definition information for a resource or a resource class by issuing the **lsrsrcdef** command with the name of a resource class.

Various command options, as shown in Table 48, enable you to display attribute definition information for the following:

- resource class or resource instances of the class
- persistent attributes or dynamic attributes

Table 48. Displaying attribute definition information for a resource or resource class

To display attribute definition information for...	Persistent attributes	Dynamic attributes
A resource class	**lsrsrcdef -c -A p -p 0 -e** *Resource_Class*	**lsrsrcdef -c -A d -p 0 -e** *Resource_Class*
A resource	**lsrsrcdef -A p -p 0 -e** *Resource_Class*	**lsrsrcdef -A d -p 0 -e** *Resource_Class*

The **lsrsrcdef** commands shown in Table 48 will return the definition for each persistent or dynamic attribute of the requested resource class or resource.

Example: The following command output shows the attribute definition returned for two attributes of the IBM.Host resource. The **-e** flag specifies expanded output format. The expanded format includes the `description` field.

```
attribute 7:
        program_name  = "OSName"
        display_name  = "Operating System Name"
        group_name    = "General"
        properties    = {"read_only","inval_for_define","selectable","public"}
        description   = "This attribute reflects the name of the operating syste
m running on the node (e.g. Linux, AIX, ...)."
        attribute_id  = 6
        group_id      = 0
        data_type     = "char_ptr"
        variety_list  = {[1,5]}
        variety_count = 1
        default_value = ""
attribute 8:
        program_name  = "KernelVersion"
        display_name  = "Kernel Version"
```

```
        group_name    = "General"
        properties    = {"read_only","inval_for_define","selectable","public"}
        description   = "This attribute reflects the version of the operating sy
stem kernel running
on the node."
        attribute_id  = 7
        group_id      = 0
        data_type     = "char_ptr"
        variety_list  = {[1,5]}
        variety_count = 1
        default_value = ""
```

If you want to return the definition of specific attributes only, include the attribute name(s) on the **lsrsrcdef** command line.

Example: `lsrsrcdef -e IBM.Host KernelVersion`

Targeting nodes:

The **lsrsrcdef** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **lsrsrcdef** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope. If the CT_MANAGEMENT_SCOPE environment variable is not set and the **-a** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lsrsrcdef** command man page and "Target nodes for a command" on page 125.

For complete syntax information on the **lsrsrcdef** command, see its online man page.

Related tasks:

"Defining a new tiebreaker" on page 91
In addition to the predefined tiebreakers, you can also create your own tiebreaker. To do so, use the **mkrsrc** command to define a new **IBM.TieBreaker** resource.

Basic resource monitoring

You can use Event Response Resource Manager commands to monitor your system of cluster domains.

You can monitor events of interest (called *conditions*) and have the RMC system react in particular ways (called *responses*) if the event occurs. To do this you create a condition/response association using the **mkcondresp** command, and then issue the **startcondresp** command to start monitoring the condition. Using the CT_MANAGEMENT_SCOPE environment variable, you can determine the set of nodes that will be monitored — either the local node only, the nodes in a peer domain, or the nodes in a management domain.

Basic resource monitoring—that is, monitoring using only predefined conditions and responses—involves the following tasks:

- Listing conditions, responses, and condition/response associations using the **lscondition**, **lsresponse**, and **lscondresp** commands.
- Creating a condition/response association using the **mkcondresp** command.
- Starting condition monitoring using the **startcondresp** command.
- Stopping condition monitoring using the **stopcondresp** command.
- Removing a condition/response association using the **rmcondresp** command.

For information on creating your own conditions, compound conditions, and responses rather than using the predefined ones provided by the various resource managers, see “Advanced resource monitoring” on page 152.

For detailed syntax information on any the commands discussed here, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Related concepts:

“Resource managers provided with RSCT” on page 110
Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

Listing conditions, responses, and condition/response associations

There are three commands for listing condition and response information. These are useful when working with conditions, responses, and condition/response associations.

These commands are:

- **lscondition** for listing information about conditions.
- **lsresponse** for listing information about responses.
- **lscondresp** for listing information about condition/response associations.

Listing conditions:

For a list of all available conditions, enter the **lscondition** command.

For example, enter the following at the command prompt:

```
lscondition
```

The following results are displayed:

```
 Name                       Node          MonitorStatus
"FileSystem space used"    "nodeA"       "Monitored"
"tmp space used"           "nodeA"       "Not monitored"
"var space used"           "nodeA"       "Not monitored"
```

Results will differ depending on what resource managers are available. The list will include any predefined conditions provided by the various resource managers, and also any conditions you create (as described in “Creating a condition” on page 157). The `MonitorStatus` in the preceding output indicates whether or not the condition is currently being monitored.

To list more detailed information about a particular condition, specify its name as a parameter to the **lscondition** command.

Example: To get detailed information about the `"FileSystem space used"` condition, enter the following at the command prompt:

```
lscondition "FileSystem space used"
```

The following results are displayed:

```
Name             = "FileSystem space used"
Location         = "nodeA"
MonitorStatus    = "Monitored"
ResourceClass    = "IBM.FileSystem"
EventExpression  = "PercentTotUsed > 99"
EventDescription = "Generate event when space used is
                    greater than 99 percent full"
RearmExpression  = "PercentTotUsed < 85"
RearmDescription = "Start monitoring again after it is
                    less than 85 percent"
```

```
SelectionString  = ""
Severity         = "w"
NodeNamelist     = "{}"
MgtScope         = "l"
Toggle           = "yes"
```

Targeting nodes:

The **lscondition** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **lscondition** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope. If the CT_MANAGEMENT_SCOPE environment variable is not set and the **-a** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lscondition** command man page and "Target nodes for a command" on page 125.

For detailed syntax information on the **lscondition** command, see its online man page.

Listing responses:

For a list of all available responses, enter the **lsresponse** command.

For example, enter the following at the command prompt:

```
lsresponse
```

The following results are displayed:

```
ResponseName
"E-mail root anytime"
"E-mail root first shift"
"Critical notifications"
"Generate SNMP trap"
```

Results will differ depending on what resource managers are available. The list will include any predefined responses provided by the various resource managers, and also any responses you create (as described in "Creating a response" on page 168).

To list more detailed information about a particular response, specify its name as a parameter to the **lsresponse** command.

Example: To get detailed information about the `"Informational notifications"` response, enter the following at the command prompt:

```
lsresponse "Informational notifications"
```

This displays the following output showing details for the two actions associated with this response.

```
Displaying response information:

        ResponseName    = "Informational notifications"
        Node            = "c175n06.ppd.pok.ibm.com"
        Action          = "Log info event"
        DaysOfWeek      = 1-7
        TimeOfDay       = 0000-2400
        ActionScript    = "/usr/sbin/rsct/bin/logevent /tmp/infoEvents"
        ReturnCode      = -1
        CheckReturnCode = "n"
        EventType       = "b"
        StandardOut     = "n"
```

```
EnvironmentVars = ""
UndefRes        = "n"

ResponseName    = "Informational notifications"
Node            = "c175n06.ppd.pok.ibm.com"
Action          = "E-mail root"
DaysOfWeek      = 2-6
TimeOfDay       = 0800-1700
ActionScript    = "/usr/sbin/rsct/bin/notifyevent root"
ReturnCode      = -1
CheckReturnCode = "n"
EventType       = "b"
StandardOut     = "n"
EnvironmentVars = ""
UndefRes        = "n"
```

Targeting nodes:

The **lsresponse** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **lsresponse** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope. If the CT_MANAGEMENT_SCOPE environment variable is not set and the **-a** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lsresponse** command man page and “Target nodes for a command” on page 125.

For detailed syntax information on the **lsresponse** command, see its online man page.

Listing condition/response associations:

Many predefined conditions and responses are provided by the various resource managers on your system. What's more, you can create your own conditions, compound conditions, and responses.

As described in “Listing conditions” on page 137 and “Listing responses” on page 138, many predefined conditions and responses are provided by the various resource managers on your system. What's more, you can create your own conditions, compound conditions, and responses as described in “Advanced resource monitoring” on page 152. Before you can monitor a condition or compound condition, however, you must link it with one or more responses. This is called a condition/response association, and is required for monitoring so that RMC knows how to respond when the condition event or compound condition event occurs.

For a list of all available condition/response associations, enter the **lscondresp** command.

Example: If no condition/response associations have been created, entering the following at the command prompt:

```
lscondresp
```

Results in the output:

```
lscondresp: No defined condition-response links were found
```

Once you link conditions and compound conditions with responses (as described in “Creating a condition/response association” on page 140), entering the **lscondresp** command will show the associations.

Example: The following **lscondresp** output is displayed:

```
Condition                 Response                      Node      State         Type
"FileSystem space used"   "Broadcast event on-shift"    "nodeA"   "Active"      c
"FileSystem space used    "E-mail root anytime"         "nodeA"   "Not Active"  c
"Page in Rate"            "Log event anytime"           "nodeA"   "Active"      c
"Paging and Idle Low"     "E-mail root anytime"         "nodeA"   "Active"      cc
```

The **Type** indicates whether the **Condition** is a condition (c) or a compound condition (cc). (Compound conditions are discussed later in "Advanced resource monitoring" on page 152.)

If you want to list the condition/response associations for a single condition or compound condition, supply the name of the condition or compound condition as a parameter to the **lscondresp** command.

Example: To list the condition/response associations for the "FileSysem space used" condition, you would enter the following command:

```
lscondresp "FileSystem space used"
```

The following output is displayed:

```
Condition                 Response                      Node      State         Type
"FileSystem space used"   "Broadcast event on-shift"    "nodeA"   "Active"      c
"FileSystem space used    "E-mail root anytime"         "nodeA"   "Not Active"  c
```

Example: If you wanted to limit the preceding output to show just the active condition/response associations, you would use the **lscondresp** command's **-a** option, as follows:

```
lscondresp -a "FileSystem space used"
```

Output would show only the active condition/response associations for the "FileSysem space used" condition.

```
Condition                 Response                      Node      State         Type
"FileSystem space used"   "Broadcast event on-shift"    "nodeA"   "Active"      c
```

Targeting nodes:

The **lscondresp** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **lscondresp** command's **-z** flag, if specified, indicates that the command applies to all nodes in the management scope. If the CT_MANAGEMENT_SCOPE environment variable is not set and the **-z** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lscondresp** command man page and "Target nodes for a command" on page 125.

For detailed syntax information on the **lscondresp** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Creating a condition/response association

Before you can monitor a condition or compound condition, you must link it with one or more responses. This is called a condition/response association and is required for monitoring so that RMC knows how to respond when the condition event or compound condition event occurs.

Many predefined conditions and responses are provided by the various resource managers on your system. What's more, you can create your own conditions, compound conditions, and responses as described in "Advanced resource monitoring" on page 152. (In the remainder of this topic, statements about conditions equally apply to compound conditions. Compound conditions provide the capability to execute response scripts when multiple conditions meet some specified criteria.)

"Listing conditions, responses, and condition/response associations" on page 137 described how to list the available conditions and responses, as well as any existing condition/response associations. You will need to know that information as you set out to create new condition/response associations.

To create an association between a condition and one or more responses, use the **mkcondresp** command. The **mkcondresp** command links responses with a condition, but does not start monitoring the condition. (Monitoring is discussed in "Starting condition monitoring" on page 142.)

Example: To use the **mkcondresp** command to link the condition `"FileSystem space used"` with the response `"Broadcast event on-shift"`, enter the following command:

```
mkcondresp "FileSystem space used" "Broadcast event on-shift"
```

You can also specify multiple responses that you want to associate with a condition.

Example: The following command links both the `"Broadcast event on-shift"` and `"E-mail root any time"` responses with the `"FileSystem space used"` condition:

```
mkcondresp "FileSystem space used" "Broadcast event on-shift" "E-mail root any time"
```

When monitoring in a management domain or peer domain scope, the condition and response you link must be defined on the same node. By default, the **mkcondresp** command assumes this to be the local node. If they are defined on another node, you can specify the node name along with the condition.

Example: The following command links the `"Broadcast event on-shift"` response with the `"FileSystem space used"` condition on node *nodeA*:

```
mkcondresp "FileSystem space used":nodeA "Broadcast event on-shift"
```

Although you specify the node name on the condition, be aware that *both* the condition and response must be defined on that node.

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server. For more information, see the **mkcondresp** command man page and "Target nodes for a command" on page 125.

Once you have linked one or more responses with a condition using the **mkcondresp**, you can verify that the condition/response association has been created by issuing the **lscondresp** command (as described in "Listing condition/response associations" on page 139).

The **mkcondresp** command links responses with a condition, but does not start monitoring the condition. To start monitoring the condition, use the **startcondresp** command, described in "Starting condition monitoring" on page 142.

To prevent user modification or removal of a condition/response link, you can lock it (as described in "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194).

For detailed syntax information on the **mkcondresp** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Starting condition monitoring

The **startcondresp** command starts monitoring a condition or compound condition that has one or more linked responses.

If you have already created these condition/response associations using the **mkcondresp** command (as described in "Creating a condition/response association" on page 140), you can specify the name of the condition or compound condition that you want to start monitoring as a parameter of the **startcondresp** command. (In the remainder of this topic, statements about conditions equally apply to compound conditions. Compound conditions provide the capability to execute response scripts when multiple conditions meet some specified criteria.)

Example: The following command starts monitoring the condition `"FileSystem space used"` using all of its linked responses:

```
startcondresp "FileSystem space used"
```

For a list of existing condition/response associations, you can use the **lscondresp** command as described in "Listing condition/response associations" on page 139. The listing returned by **lscondresp** also shows the state of the condition/response association (active or not active), so you can use it to verify that monitoring has started.

If a condition has multiple linked responses but you do not want RMC to use all of them, you can explicitly state which response you want triggered when the condition is true. You do this by specifying the responses as parameters to the **startcondresp** command.

Example: If the `"FileSystem space used"` condition has multiple responses linked with it, you could issue the following command to start monitoring that will use only the `"Broadcast event on-shift"` response:

```
startcondresp "FileSystem space used" "Broadcast event on-shift"
```

If you wanted to also use the `"E-mail root any time"` response, you would enter:

```
startcondresp "FileSystem space used" "Broadcast event on-shift" "E-mail root anytime"
```

You can also use the above format of specifying a response on the **startcondresp** command to create a condition/response association and start monitoring in one step.

Example: If the `"FileSystem space used"` condition had not already been linked with the `"Broadcast event on-shift"` response (using the **mkcondresp** command), then the command:

```
startcondresp "FileSystem space used" "Broadcast event on-shift"
```

would both create the association and start monitoring. In this way, the **startcondresp** command is like the **mkcondresp** command. The difference is that the **mkcondresp** command merely creates the condition/response association, while the **startcondresp** command creates the association and starts monitoring.

If using the **startcondresp** command to create a command/response association, be aware that, when monitoring in a management domain or peer domain scope, the condition and response you link must be defined on the same node. By default, the **startcondresp** command assumes this to be the local node. If they are defined on another node, you can specify the node name along with the condition.

Example: The following command starts monitoring the `"FileSystem space used"` condition using the `"Broadcast event on-shift"` response (and creates an association, if they were not already linked) on node *nodeA*:

```
startcondresp "FileSystem space used":nodeA "Broadcast event on-shift"
```

Although you specify the node name on the condition, be aware that *both* the condition and response must be defined on that node.

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server. For more information, see the **startcondresp** command man page and "Target nodes for a command" on page 125.

If you start monitoring for a compound condition, the conditions that the compound condition refers to are also started, if they have not yet been started.

To prevent a user from stopping monitoring, you can lock the condition/response link (as described in "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194). Locking a condition/response link also prevents accidental removal of the link.

For detailed syntax information on the **startcondresp** command, see its online man page.

Stopping condition monitoring

The **stopcondresp** command stops monitoring of a condition or compound condition that has one or more linked responses.

In this topic, statements about conditions equally apply to compound conditions. Compound conditions provide the capability to execute response scripts when multiple conditions meet some specified criteria.

Example: To stop all active responses for the `"FileSystem space used"` condition, enter the following command:

```
stopcondresp "FileSystem space used"
```

If you are unsure which conditions are currently being monitored, you can use the **lscondition** command as described in "Listing conditions" on page 137.

If the condition has multiple linked and active responses and you only want to stop some of those responses while allowing the others to remain active, simple specify the response(s) you want to deactivate as parameters on the **stopcondresp** command. (To ascertain which responses are active for the condition, use the **lscondresp** command, as described in "Listing conditions" on page 137.)

Example: If you want to deactivate the `"Broadcast event on-shift"` response for the `"FileSystem space used"` condition, enter the following command:

```
stopcondresp "FileSystem space used" "Broadcast event on-shift"
```

Example: If you want to deactivate the responses `"Broadcast event on-shift"` and `"E-mail root anytime"` for the `"FileSystem space used"` condition, enter:

```
stopcondresp "FileSystem space used" "Broadcast event on-shift" "E-mail root anytime"
```

If the condition you want to stop monitoring is defined on another node, you can specify the node name along with the condition.

Example: The following command stops monitoring the `"FileSystem space used"` condition using the `"Broadcast event on-shift"` response on node *nodeA*:

```
stopcondresp "FileSystem space used":nodeA "Broadcast event on-shift"
```

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server. For more information, see the **stopcondresp** command man page and "Target nodes for a command" on page 125.

If the condition/response association you specify on the **stopcondresp** command is locked, monitoring will not be stopped; instead, an error will be generated informing you that the condition/response link is locked. For information on unlocking a condition/response link so monitoring can be stopped, see "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

For detailed syntax information on the **stopcondresp** command, see its online man page.

Removing a condition/response association

The **rmcondresp** command removes the association between a condition or compound condition and one or more responses.

To see a list of the existing condition/response associations that you can remove, you can use the **lscondresp** command as described in "Listing condition/response associations" on page 139. The **rmcondresp** command enables you to remove a specified condition/response association, all the associations for a specified condition (or compound condition) or all the associations for a specified response. (In the remainder of this topic, statements about conditions equally apply to compound conditions. Compound conditions provide the capability to execute response scripts when multiple conditions meet some specified criteria.)

To remove a specific condition/response association, specify both the condition and response as parameters to the **rmcondresp** command.

Example: The following command removes the condition/response association between the `"FileSystem space used"` condition and the `"Broadcast event on-shift"` response:

```
rmcondresp "FileSystem space used" "Broadcast event on-shift"
```

You can also delete the links between a condition and multiple responses.

Example: The following command removes the associations between the `"FileSystem space used"` condition and the responses `"Broadcast event on-shift"` and `"E-mail root anytime"`:

```
rmcondresp "FileSystem space used" "Broadcast event on-shift" "E-mail root any time"
```

To remove links to all responses associated with a particular condition, specify the condition only as a parameter to the **rmcondresp** command.

Example: To remove the links to all responses associated with the `"FileSystem space used"` condition, enter the following command:

```
rmcondresp "FileSystem space used"
```

Similarly, you can remove condition/response associations from all conditions that are linked to one or more responses by using the **-r** option. The **-r** option tells the **rmcondresp** command that all the command parameters are responses.

Example: The following command removes all condition/response associations that use the `"Broadcast event on-shift"` response:

```
rmcondresp -r  "Broadcast event on-shift"
```

You can also specify multiple responses with the **-r** option.

Example: The following example removes all condition/response associations that use the `"Broadcast event on-shift"` or `"E-mail root any time"` responses:

```
rmcondresp -r "Broadcast event on-shift" "E-mail root any time"
```

If the condition and response you want to stop monitoring are defined on another node, you can specify the node name along with the condition.

Example: The following command removes the condition/response association between the `"FileSystem space used"` condition and the `"Broadcast event on-shift"` response on *nodeA*:

```
rmcondresp "FileSystem space used":nodeA "Broadcast event on-shift"
```

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server. For more information, see the **rmcondresp** command man page and "Target nodes for a command" on page 125.

If the condition/response link you specify on the **rmcondresp** command is locked, it will not be removed; instead, an error will be generated informing you that the condition/response link is locked. For information on unlocking the condition/response link so it can be removed, see "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

For detailed syntax information on the **rmcondresp** command, see its online man page.

Using the audit log to track monitoring activity

When you are monitoring a condition or compound condition, be aware that any linked response actions will be executed in the background by daemons.

Often, the response action will somehow log or notify you about the event occurring. For example, RSCT's predefined responses use response scripts that do one of the following:

- log information to a file
- mail the information to a particular user ID
- broadcast the information to all users who are logged in

In some cases, you might create your own response script that performs no such logging or notification but, instead, provides a more targeted solution for when the monitored attribute tests True. For example, you might create a recovery script that deletes unnecessary files when the **/tmp** directory is 90% full.

Whether or not the response script performs some type of notification or logging itself, it is important to know that RMC has an audit log in which it records information about the system's operation and that the event response resource manager appends entries to this log for all triggered response actions. The audit log includes information about the normal operation of the system as well as failures and other errors and, thus, augments any information that a response script might provide.

You can use the **lsevent** and **lsaudrec** commands to track monitoring activity. The **lsevent** command is described in "Listing event monitoring information from the audit log" on page 146 and the **lsaudrec** command is described in "Listing records from the audit log" on page 147.

Related concepts:

"Resource managers provided with RSCT" on page 110
Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

Listing event monitoring information from the audit log:

The **lsevent** command lists event monitoring information recorded by the Event Response resource manager in the audit log.

Statements about conditions equally apply to compound conditions. Compound conditions provide the capability to execute response scripts when multiple conditions meet some specified criteria.

Without any operands, **lsevent** lists the events that are recorded in the audit log—these describe the monitored events that have occurred.

Example: To list the information for events that have occurred, enter:

```
lsevent
```

You can specify a condition name (or compound condition name) to list events for a particular condition.

Example: To list event information about the `"FileSystem space used"` condition, enter:

```
lsevent "FileSystem space used"
```

Response information can be listed separately or with the event information. Responses are run based on a condition or event occurring. Information about a response includes when it was run, what the response script was, the return code, the expected return code (if the response was defined so as to record it), standard error output, and standard output (if the response was defined so as to record it).

If you want to list the event responses for a condition or both the events and event responses for a condition, specify the **-R** or **-A** flag, respectively. You can also specify one or more response names to limit the response output.

Examples: The following examples illustrate the use of the **lsevent** command with the **-R** and **-A** flags to list information about events and event responses from the audit log.

- To list all event response information for the `"FileSystem space used"` condition, enter:

  ```
  lsevent -R "FileSystem space used"
  ```

- To list only the event responses for the `"Broadcast event on-shift"` response for the `"FileSystem space used"` condition, enter:

  ```
  lsevent -R "FileSystem space used" "Broadcast event on-shift"
  ```

- To list both event information and event response information for the `"Broadcast event on-shift"` response for the `"FileSystem space used"` condition, enter:

  ```
  lsevent -A "FileSystem space used" "Broadcast event on-shift"
  ```

You can use the **-r** flag to list information about event responses. The **-r** flag tells **lsevent** that all command parameters, if any, are response names and that event response information is to be listed for the specified response names. If no response names are specified along with the **-r** flag, then information for all event responses is listed.

Examples: The following examples illustrate the use of the **lsevent** command with the **-r** flag to list event response information from the audit log.

- To list all event response information, enter:

  ```
  lsevent -r
  ```

- To list event response information for the `"Broadcast event on-shift"` response, enter:

```
lsevent -r "Broadcast event on-shift"
```

You can also limit the portion of the audit log that is searched by specifying a beginning timestamp (using the **-B** flag), an ending timestamp (using the **-E** flag), or both, and by specifying the number of most recent records to be searched (using the **-O** flag). You can use these flags in combination with any of the other event and event response criteria discussed above.

Examples: The following examples illustrate the use of the **lsevent** command with the **-O**, **-B**, and **-E** flags to list event information from a specific portion of the audit log.

- To see event information for the `"FileSystem space used"` condition found in the latest 1000 records in the audit log, enter:

  ```
  lsevent -O 1000 "FileSystem space used"
  ```

- To see event information for the `"FileSystem space used"` condition that occurred on July 27th between 14:30 and 15:00, enter:

  ```
  lsevent -B 072714302006 -E 072715002006 "FileSystem space used"
  ```

  The timestamps are in the form `MMddhhmmyyyy`, where `MM` = month, `dd` = day, `hh` = hour, `mm` = minutes, and `yyyy` = year. The timestamp can be truncated from right to left, except for `MM`. If not present, the following defaults are used:
  - year = the current year
  - minutes = `00`
  - hour = `00`
  - day = `01`

Targeting nodes:

The **lsevent** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope.

The **lsevent** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope.

The **lsevent** command's **-n** flag specifies a list of nodes containing the audit log records to display. Any node specified must within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set).

If the CT_MANAGEMENT_SCOPE environment variable is not set and either the **-a** flag or **-n** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lsevent** command man page and “Target nodes for a command” on page 125.

For detailed syntax information on the **lsevent** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Listing records from the audit log:

To list records from the audit log, use the **lsaudrec** command.

For example. to list all records in the audit log, enter:

```
lsaudrec
```

The following output is displayed:

```
Time              Subsystem Category Description
07/27/02 14:55:42      ERRM Info     Monitoring of condition Processor idle time
 is started successfully.
07/27/02 14:55:58      ERRM Info     Event : Processor idle time occurred at 07/
27/02 14:55:58 953165 on proc0 on c175n06.ppd.pok.ibm.com.
07/27/02 14:55:59      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 will cause /usr/sbin/rsct/bin/logevent /tmp/system
Events from Log event anytime to be executed.
07/27/02 14:55:59      ERRM Info     Event : Processor idle time occurred at 07/
27/02 14:55:58 953165 on proc1 on c175n06.ppd.pok.ibm.com.
07/27/02 14:55:59      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 will cause /usr/sbin/rsct/bin/logevent /tmp/system
Events from Log event anytime to be executed.
07/27/02 14:55:59      ERRM Info     Event : Processor idle time occurred at 07/
27/02 14:55:58 953165 on proc2 on c175n06.ppd.pok.ibm.com.
07/27/02 14:55:59      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 will cause /usr/sbin/rsct/bin/logevent /tmp/system
Events from Log event anytime to be executed.
07/27/02 14:55:59      ERRM Info     Event : Processor idle time occurred at 07/
27/02 14:55:58 953165 on proc3 on c175n06.ppd.pok.ibm.com.
07/27/02 14:55:59      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 will cause /usr/sbin/rsct/bin/logevent /tmp/system
Events from Log event anytime to be executed.
07/27/02 14:56:00      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 caused /usr/sbin/rsct/bin/logevent /tmp/systemEven
ts from Log event anytime to complete with a return code of 0.
07/27/02 14:56:00      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 caused /usr/sbin/rsct/bin/logevent /tmp/systemEven
ts from Log event anytime to complete with a return code of 0.
07/27/02 14:56:00      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 caused /usr/sbin/rsct/bin/logevent /tmp/systemEven
ts from Log event anytime to complete with a return code of 0.
07/27/02 14:56:00      ERRM Info     Event from Processor idle time that occurre
d at 07/27/02 14:55:58 953165 caused /usr/sbin/rsct/bin/logevent /tmp/systemEven
ts from Log event anytime to complete with a return code of 0.
07/27/02 14:56:51      ERRM Info     Monitoring of condition Processor idle time
 is stopped sucessfully.
```

The above example shows:

- When RMC started monitoring the "`Processor idle time`" condition
- Each time the "`Processor idle time`" condition tested true
- That the "`Log event anytime`" response was associated with the "`Processor idle time`" condition and, as a result, its response action "`/usr/sbin/rsct/bin/logevent /tmp/systemEvents`" was executed each time the condition tested true
- The return code from each execution of the command "`/usr/sbin/rsct/bin/logevent /tmp/systemEvents`"
- When RMC stopped monitoring the "`Processor idle time`" condition

The above audit log is quite small and contains entries related to a single monitored condition. In practice, however, the audit log is likely to contain a very large number of records. For this reason, the **lsaudrec** command enables you to filter the audit log so that only a subset of its records are returned.

To filter the audit log, use the **lsaudrec** command's **-s** option followed by a *selection string* — an expression that determines how the audit log is to be filtered. Every record in the audit log has a number of named fields (such as **Time**) the provide specific information associated with the record. These field names are used in the selection string expression, which also includes constants and operators. Expressions in RMC are discussed in more detail in “Using expressions to specify condition events and command selection strings” on page 196. Here, it suffices to say that the syntax of the selection string is similar to an expression in the C programming language or the *where* clause in SQL. The selection string you provide is matched against each record in the audit log. The **lsaudrec** man page contains detailed

syntax information on the **-s** option and the field names you can use when filtering the audit log. Here, we will discuss only the most common field names you would typically use when filtering the audit log.

Example: It is common to want to filter the audit log based on the time that records were created. You can do this using the **-s** flag and the **Time** field name. To filter the audit log so that only records created on July 27 between 14:30 and 15:00 are listed, you would enter the following command:

```
lsaudrec -s "Time > #072714302006 && Time < #072715002006"
```

The expression used in the preceding example specifies the date/time using the format `#mmddhhmmyyyy`, where, from left to right: `mm` = month, `dd` = day, `hh` = hour, `mm` = minutes, and `yyyy` = year. The fields can be omitted from right to left. If not present, the following defaults are used:

- year = the current year
- minutes = 00
- hour = 00
- day = 01
- month = the current month

Example: To issue the same command as in the previous example but using the current year as the default, enter:

```
lsaudrec -s "Time > #07271430 && Time < #07271500"
```

You can also specify the time using the format `#-mmddhhmmyyyy`. In this case, the time specified is relative to the current time. Again, fields can be omitted from right to left; for this format the omitted fields are replaced by 0 So, for example, the value `#-0001` corresponds to one day ago, and the value `#-010001` corresponds to one month and one hour ago. To list the audit log entries that were logged in the last hour only, you would enter:

```
lsaudrec -s "Time > #-000001"
```

Another field that is commonly used when filtering the audit log is the **Category** field. If the **Category** field of an audit log record is 0, it is an informational message. If the **Category** field of an audit log record is 1, it is an error message.

Example: To list just the error messages in an audit log, enter:

```
lsaudrec -s "Category=1"
```

Targeting nodes:

The **lsaudrec** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope.

The **lsaudrec** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope.

The **lsaudrec** command's **-n** flag specifies a list of nodes containing the audit log records to display. Any node specified must within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set).

If the CT_MANAGEMENT_SCOPE environment variable is not set and either the **-a** flag or **-n** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **lsaudrec** command man page and “Target nodes for a command” on page 125.

For detailed syntax information on the **lsaudrec** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Deleting entries from the audit log:

There are two ways to delete entries from the audit log — explicitly (using the **rmaudrec** command) or implicitly (by setting the **RetentionPeriod** and **MaxSize** attributes of the IBM.AuditLog resource).

Deleting entries from the audit log using the rmaudrec command:

The**rmaudrec** command removes records from the audit log.

You must provide this command with a *selection string* — an expression that indicates which records should be deleted. Like the **lsaudrec** command, the **rmaudrec** command has an **-s** option for specifying the selection string expression, which takes the same form as it does on the **lsaudrec** command. For example, to remove all records from the audit log, you would enter:

```
rmaudrec -s "Time > 0"
```

To remove only the records that were created on July 27 between 14:30 and 15:00, you would enter:

```
rmaudrec -s "Time > #07271430 && Time < #07271500"
```

To delete the audit log entries that were logged in the last hour only, you would enter:

```
rmaudrec -s "Time > #-000001"
```

To remove only informational messages from the audit log (leaving error messages), you would enter:

```
rmaudrec -s "Category=0"
```

Targeting nodes:

The **rmaudrec** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope.

The **rmaudrec** command's **-a** flag, if specified, indicates that the command applies to all nodes in the management scope.

The **rmaudrec** command's **-n** flag specifies a list of nodes whose audit log records can be deleted (if they meet other criteria such as matching the selection string). Any node specified must be defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set).

If the CT_MANAGEMENT_SCOPE environment variable is not set and either the **-a** flag or **-n** flag is specified, then the default management scope will be the management domain scope if it exists. If it does not, then the default management scope is the peer domain scope if it exists. If it does not, then the management scope is the local scope. For more information, see the **rmaudrec** command man page and “Target nodes for a command” on page 125.

For detailed syntax information on the **lsaudrec** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Deleting entries from the audit log using the IBM.AuditLog resource's RetentionPeriod and MaxSize attributes:

In addition to being able to explicitly delete audit log entries using the **rmaudlog** command, you can also set certain attributes of the IBM.AuditLog resource that represents the audit log so that RMC will automatically delete records from the audit log.

These attributes are:

- the **RetentionPeriod** attribute which determines how many hours RMC should keep records in the audit log. Records older than the number of hours indicated are automatically deleted by RMC. If the **RetentionPeriod** attribute value is set to 0, this indicates that audit log records should not be automatically deleted based on their age.
- the **MaxSize** attribute which determines the maximum size (in megabytes) of the audit log. If the size of the audit log exceeds the size indicated, RMC will automatically remove the oldest records until the size of the audit log is smaller that the indicated limit. The default size limit of the audit log is 1 megabyte.

To list the current attribute settings, use the **lsrsrc** command (described in more detail in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides). To list the attribute settings for the IBM.AuditLog instance that represents the ERRM audit log, use the selection string `-s 'Name == "ERRM"'`. This selection string is necessary since other subsystems may have their own audit logs.

Example:

```
lsrsrc -s 'Name == "ERRM"' IBM.AuditLog
```

The following output is displayed:

```
Resource Persistent Attributes for: IBM.AuditLog
resource 1:
        Name            = "ERRM"
        MessageCatalog  = "IBM.ERrm.cat"
        MessageSet      = 1
        DescriptionId   = 38
        DescriptionText = "This subsystem is defined by ERRM for recording signi
ficant event information."
        RetentionPeriod = 0
        MaxSize         = 1
        SubsystemId     = 1
        NodeNameList    = {"c175n06.ppd.pok.ibm.com"}
```

Included in this output are the attribute settings for the **RetentionPeriod** and **MaxSize** attributes. The **RetentionPeriod** attribute is set to 0; this indicates that RMC should not automatically delete records based on their age. The **MaxSize** attribute is set to 1; RMC will automatically delete the oldest records from the audit log when the audit log size exceeds 1 megabyte.

To change these settings, use the **chrsrc** command.
Examples:

1. To specify that RMC should automatically delete records that are over a day old, you would set the **RetentionPeriod** attribute as follows:

   ```
   chrsrc -s 'Name == "ERRM"' IBM.AuditLog RetentionPeriod=24
   ```

2. To increase the maximum size of the audit log to 10 megabytes, you would enter:

   ```
   chrsrc -s 'Name == "ERRM"' IBM.AuditLog MaxSize=10
   ```

Note: The default size limit of the audit log is 1 megabyte, which will be an insufficient size for a large cluster. In a large cluster, you will likely want to increase the audit log size as shown in the preceding example. If you do set the **MaxSize** attribute to increase the maximum size limit of the audit log, be sure to verify that the size of the file system containing the log (by default, the **/var** file system) has enough

room to hold it. Because RSCT subsystems use the **/var** file system extensively, it is also a good idea to monitor its size. To monitor the **/var** file system, you can use the predefined condition **/var space used** that is provided by the file system resource manager.
For more information about the **chrsrc** and **lsrsrc** commands, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Advanced resource monitoring

Many predefined conditions and responses are provided by the various resource managers on your system. These predefined conditions and responses are provided as an administrative convenience.

As described in "Basic resource monitoring" on page 136, many predefined conditions and responses are provided by the various resource managers on your system. These predefined conditions and responses are provided as an administrative convenience. As described in "Creating a condition/response association" on page 140, you can use them to create condition/response associations for monitoring. However, the predefined conditions and responses may not always meet your needs. In such cases, you can use advanced resource monitoring. Advanced resource monitoring includes:

- Creating your own conditions that can then be linked with one or more responses and monitored by RMC. If the condition you wish to monitor is similar to one of the predefined conditions available on your system, you can copy the existing condition and modify it as needed. If none of the existing conditions are similar to the condition you want to monitor, you can create a new condition from scratch. This involves identifying the attribute you want to monitor for one or more resources of a particular resource class. Since persistent attributes are generally unchanging, you will usually monitor a dynamic attribute. If none of the dynamic attributes provided by the resource managers contains the value you want to monitor, you can create a *sensor*—a command to be run by RMC to retrieve the value you want to monitor. For more information, see "Creating, modifying and removing conditions" on page 154.
- Creating your own compound conditions that can then be linked with one or more responses and monitored by RMC. If the compound condition you wish to monitor is similar to one that already exists on your system, you can copy the existing compound condition and modify it as needed. If none of the existing compound conditions is similar to the compound condition you want to monitor, you can create a compound condition from scratch. This involves identifying the existing conditions whose event notifications you want to monitor, correlate, and evaluate.
- Creating your own responses that can then be linked with conditions. You can use predefined response scripts in your responses and you can also create your own response scripts. For more information, see "Creating, modifying, and removing responses" on page 166.

Once you know how to create conditions, compound conditions, and responses, be aware that, to be effective, you need to link the conditions or compound conditions together with one or more responses and start monitoring. These tasks are described in "Creating a condition/response association" on page 140 and "Starting condition monitoring" on page 142. For detailed syntax information on any of the commands, see *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms*.

Tuning reporting intervals

You can adjust reporting intervals for resource dynamic attributes. A *reporting interval* is the amount of time between a resource manager's sampling of values. You can set reporting intervals for all classes, a single class, or a single attribute of a class.

To adjust the reporting intervals for resource dynamic attributes that are defined in a class definition file, you can create a *reporting interval override file.* This file is called:

```
/var/ct/cfg/ctrmc.rio
```

Information in this file only applies to resource dynamic attributes that have reporting intervals, which have a variable type of **Counter** or **Quantity**.

ctrmc.rio is a plain text file, in which a line in the file contains two white-space-separated tokens, as follows:

```
class_name[:attr_name] reporting_interval
```

The first token is a class name as returned by the **lsrsrc** command, or, a class name followed immediately by a colon, which is followed by an optional resource dynamic attribute name as returned by the **lsrsrcdef** command. If only a class name is specified, the reporting interval applies to all resource dynamic attributes of the class that have reporting intervals. If a class name and attribute name are specified, the reporting interval applies to the named attribute only, if appropriate. If the attribute does not have a reporting interval, the line is ignored. The second token, the reporting interval, is an unsigned integral value that is interpreted as a number of seconds. If the class name, without any attribute names, is the keyword **ALL**, the reporting interval applies to all dynamic resource attributes of all resource classes. The last specification for a given attribute applies.

In the following example:

```
Foo:larry       10
Foo             15
Foo:curly       20
```

the reporting interval would be **15** for **larry** and **20** for **curly**. All other dynamic attributes of the **Foo** class would have a reporting interval of **15**. Blank lines, lines that begin with the number sign (**#**), and lines that contain unrecognized tokens are ignored.

The reporting interval override file is read by the RMC daemon upon start. If the **refresh -s ctrmc** command is invoked, the file is re-read. In this case, any new reporting intervals apply only to future event registrations.

Related concepts:

"Resource managers provided with RSCT" on page 110
Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

Compound conditions

When a condition is monitored, it generates an event notification when its event expression evaluates to *true*. A *compound condition* provides the capability to define an event expression on one or more conditions so as to examine multiple event notifications generated by those conditions.

When a compound condition's event expression evaluates to *true*, a new event notification is generated and can be passed to one or more responses. Event notifications for the individual conditions can optionally be passed to their associated responses as well.

Just like a condition, a compound condition can be linked to, or associated with, one or more responses, monitoring of the compound condition can be started, and, when the compound condition's event expression evaluates to *true*, the associated responses can be run.

You can use the following commands to work with compound conditions:

- **lsccondition** — List information about compound conditions
- **mkccondition** — Create a new compound condition
- **chccondition** — Modify a compound condition
- **rmccondition** — Remove a compound condition

Just as you can with conditions, you can also do the following with compound conditions:

- Use the **mkcondresp** command to create an association between a compound condition and one or more responses. For more information, see "Creating a condition/response association" on page 140.

- Use the **rmcondresp** command to remove the association between a compound condition and one or more responses. For more information, see "Removing a condition/response association" on page 144.
- Use the **startcondresp** command to start monitoring a compound condition that has one or more linked responses. Starting a compound condition also starts any associated conditions, if they are not already started. For more information, see "Starting condition monitoring" on page 142.
- Use the **stopcondresp** command to stop monitoring a compound condition that has one or more linked responses. Stopping a compound condition also stops any associated conditions if they are not required to be monitored for other compound conditions or responses. For more information, see "Stopping condition monitoring" on page 143.

Locking or unlocking a compound condition:

To lock or unlock a compound condition, use the **-L** and **-U** flags on the **chccondition** command. When using either of these flags, no other operation can be performed by the **chcondition** command.

The syntax is:

chccondition {**-L** | **-U**} *compound_condition*[**:***node_name*]

Note: Specifying the management server name as the target node is not valid if you are running the command on the management server.

If you are working on the management server, the conditions will be created on the management server without specifying a target. Do not specify the management server name as the target node in any of the following commands:

- **lscondition**
- **lsresponse**
- **lscondresp**
- **mkcondresp**
- **startcondresp**
- **stopcondresp**
- **rmcondresp**
- **rmcondition**
- **chresponse**

Examples:

- If you have created a compound condition named `Low paging and Idle` and now want to lock it to prevent accidental modification or removal, you would enter:
  ```
  chccondition -L "Low paging and Idle"
  ```
- To unlock this compound condition, you would enter:
  ```
  chccondition -U "Low paging and Idle"
  ```

Creating, modifying and removing conditions

There are three commands you can use to manipulate conditions.

You can:

- Create a new condition using the **mkcondition** command.
- Modify a condition using the **chcondition** command.
- Remove a condition using the **rmcondition** command.

Before we discuss these commands, it is important that you understand the basic attributes of a condition. In "Listing conditions" on page 137, we discuss the **lscondition** command that enables you to

list conditions that are available. This command lists the predefined conditions we provide, as well as any you define. Specifying the name of a condition as a parameter to the **lscondition** command returns detailed information about the condition. For example, entering this command:

```
lscondition "/var space used"
```

Returns the following information about the predefined condition "/var space used".

```
Displaying condition information:

condition 1:
        Name             = "/var space used"
        Node             = "c175n06.ppd.pok.ibm.com"
        MonitorStatus    = "Not monitored"
        ResourceClass    = "IBM.FileSystem"
        EventExpression  = "PercentTotUsed > 90"
        EventDescription = "An event will be generated when more than 90 percent
 of the total space in the /var directory is in use."
        RearmExpression  = "PercentTotUsed < 75"
        RearmDescription = "The event will be rearmed when the percent of the sp
ace used in the /var directory falls below 75 percent."
        SelectionString  = "Name == \"/var\""
        Severity         = "i"
        NodeNames        = {}
        MgtScope         = "l"
        Toggle           = "Yes"
```

It is important to understand the information contained in this output, because you can set many of these values using the various flags of the **mkcondition** and **chcondition** commands. Table 49 explains each line of the **lscondition** command output shown above.

*Table 49. Explanation of **lscondition** command output*

This line of the lscondition command output...	Indicates...	Notes
`Name = "/var space used"`	The name of the condition. In this case `"/var space used"`.	Specified as a parameter of the **mkcondition** and **chcondition** commands.
`Node = "c175n06.ppd.pok.ibm.com"`	The node on which the condition is defined. This is important, because, when you create a condition/response association, both the condition and the response must reside on the same node. In this case, the `"/var space used"` condition is defined on the node `"c175n06.ppd.pok.ibm.com"`. This node information is provided only if the management scope is a peer domain scope or a management domain scope.	By default, will be the node where the **mkcondition** command runs. Can be explicitly specified using the **mkcondition** command's **-p** flag.
`MonitorStatus = "Not monitored"`	Whether or not the condition is being monitored. In this case, it is not.	See "Starting condition monitoring" on page 142 and "Stopping condition monitoring" on page 143.
`ResourceClass = "IBM.FileSystem"`	The resource class monitored by this condition. This will be the resource class that contains the attribute used in the event expression and, optionally, the rearm event expression. In this case, the resource class is the file system resource class (which contains the PercentTotUsed dynamic attribute used in the event expression and rearm event expressions).	Specified by the **-r** flag of both the **mkcondition** and **chcondition** commands.

*Table 49. Explanation of **lscondition** command output (continued)*

This line of the lscondition command output...	Indicates...	Notes
`EventExpression = "PercentTotUsed > 90"`	The event expression used in monitoring the condition. Once you link the condition with one or more responses (as described in "Creating a condition/response association" on page 140), and start monitoring (as described in "Starting condition monitoring" on page 142), RMC will periodically poll the resource class to see if this expression (in this case `"PercentTotUsed > 90"`) tests true. If it does test true, RMC will execute any response scripts associated with the condition's linked response(s). An event expression includes an attribute, a mathematical comparison symbol, and a constant. This particular expression uses the PercentTotUsed dynamic attribute which indicates the percentage of space used in a file system. When the file system is over 90 percent full, RMC generates an event, thus triggering any linked responses.	Specified by the **-e** flag of both the **mkcondition** and **chcondition** commands.
`EventDescription = "An event will be generated when more than 90 percent of the total space in the /var directory is in use."`	A description of the event expression.	Specified by the **-d** flag of both the **mkcondition** and **chcondition** commands.
`RearmExpression = "PercentTotUsed < 75"`	The rearm event expression. Once the event expression tests true, RMC will not test the event expression condition again until the rearm expression tests true. When this particular condition is monitored, for example, RMC will periodically poll the file system resource class to determine if the expression the test the event expression `"PercentTotUsed > 90"` is true. If it does, the linked responses are triggered, but, because there is a rearm event specified, RMC will then no longer test if `"PercentTotUsed > 90"` is true. If it did, the linked responses would be triggered every time RMC polled the file system resource class until the percentage of space used in the file system fell below 90 percent. If a linked response was to broadcast the information to all users who are logged in, the repeated broadcasts of the known problem would be unnecessary. Instead of this, the event expression testing true causes RMC to start testing the rearm event expression instead. Once it tests true, the condition is rearmed; in other words, the event expression is again tested. In this case, the condition is rearmed when the file system is less that 75 percent full. It is important to note that many conditions do not specify a rearm expression. When this is the case, the event expression will continue to be tested event after it tests true.	Specified by the **-E** flag of both the **mkcondition** and **chcondition** commands.
`RearmDescription = "The event will be rearmed when the percent of the sp ace used in the /var directory falls below 75 percent."`	A description of the rearm event expression.	Specified by the **-D** flag of both the **mkcondition** and **chcondition** commands.
`SelectionString = "Name == \"/var\""`	A selection string. This is an expression that determines which resources in the resource class are monitored. If a condition does not have selection string, then the condition would apply to all resources in the class. For example, if this condition did not have a selection string, the event expression would be tested against all file system resources in the file system resource class, and an event would occur if any of the file systems were over 90 percent full. However, since this selection string is defined, the condition applies only to the **/var** file system. The selection string can filter the resource class using any of its persistent attributes. In this case, that resource class is filtered using the **Name** attribute. Expressions in RMC are discussed in more detail in "Using expressions to specify condition events and command selection strings" on page 196.	Specified by the **-s** flag of both the **mkcondition** and **chcondition** commands.

*Table 49. Explanation of **lscondition** command output (continued)*

This line of the lscondition command output...	Indicates...	Notes
`Severity = "i"`	The severity of the condition. In this case, the condition is informational.	Specified by the **-S** flag of both the **mkcondition** and **chcondition** commands.
`NodeNames = {}`	The host names of the nodes where the condition is to be monitored. No hosts are named in this case. All nodes in the management scope will be monitored. For more information, see "The monitoring scope of a condition" on page 121.	Specified by the **-n** flag of both the **mkcondition** and **chcondition** commands.
`MgtScope = "l"`	The RMC scope in which the condition is monitored. In this case, the scope is the local node only. For more information, see "The monitoring scope of a condition" on page 121.	Specified by the **-m** flag of both the **mkcondition** and **chcondition** commands.
`Toggle = "Yes"`	The condition toggles between the event and rearm event. The default value is "Yes." If the value is set to "No" then the condition does not toggle between the event and rearm event.	Specified by the **--qtoggle** or **--qnotoggle** flag of both the **mkcondition** and **chcondition** commands.

Creating a condition:

To create a condition, you use the **mkcondition** command. Before creating a condition from scratch, make sure that it is truly necessary.

In other words, first check to see if any of the predefined conditions is already set up to monitor the event you are interested in. For instructions on listing the conditions already available on your system, see "Listing conditions" on page 137. If you have additional resource managers that are provided by other licensed programs, such as Tivoli System Automation, see that program's documentation for information about any additional predefined conditions. You might find a predefined condition that will monitor the exact event or a similar event in which you are interested.

Table 50 describes how to proceed when an existing condition suits your needs exactly, is very similar to what you want, or when there are no existing conditions that are similar to what you need.

Table 50. Using or copying an existing condition or creating a new condition

If...	Then...
There is a predefined condition that exactly suits your needs	You do not need to perform this advanced task; instead, see "Creating a condition/response association" on page 140 and "Starting condition monitoring" on page 142.
There is a predefined condition very similar to the event you want to monitor	Use the **mkcondition** command's **-c** flag to copy the existing condition, modifying only what you want to change to suit your needs. See "Creating a condition by copying an existing one" on page 158 for more information.
There is no predefined condition that is similar to the event you want to monitor	Define the condition completely from scratch. Examine the available resource managers to determine whether any of them define an attribute containing the value you want to monitor. If none of them do, you can extend RMC by creating a *sensor* — a command that RMC will run to retrieve the value you want to monitor. See "Creating a condition from scratch" on page 160.

Targeting nodes:

The **mkcondition** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **mkcondition** command's **-p**

flag, if specified, indicates the name of a node where the condition is defined. This must be a node within the management scope for the local node (or the node indicated by the CT_CONTACT environment variable).

If the CT_MANAGEMENT_SCOPE environment variable is not set, and the **-p** flag is used, this command will attempt to set the CT_MANAGEMENT_SCOPE environment variable to the management scope that contains the node specified on the **-p** flag. In this case, the specified node should be in the management domain or peer domain of the local node (or the node indicated by the CT_CONTACT environment variable).

If you use the **mkcondition** command on a management server and you want the condition to be defined on the management server, do not specify the **-p** flag.

For more information, see the **mkcondition** command man page and "Target nodes for a command" on page 125.

Creating a condition by copying an existing one:

If there is an existing condition very similar to the event you want to monitor, you can use the **mkcondition** command's **-c** flag to copy the existing condition, modifying only what you want to change to suit your needs.

For example, suppose that you want to monitor the **/var** file system, and generate an event when the file system is 85 percent full. Using the **lscondition** command, as described in "Listing conditions" on page 137, shows that there is a predefined condition named `"/var space used"`. To get detailed information about this predefined condition, enter the following command:

```
lscondition "/var space used"
```

Which causes the following information to be output.

```
Displaying condition information:

condition 1:
        Name             = "/var space used"
        Node             = "c175n06.ppd.pok.ibm.com"
        MonitorStatus    = "Not monitored"
        ResourceClass    = "IBM.FileSystem"
        EventExpression  = "PercentTotUsed > 90"
        EventDescription = "An event will be generated when more than 90 percent
 of the total space in the /var directory is in use."
        RearmExpression  = "PercentTotUsed < 75"
        RearmDescription = "The event will be rearmed when the percent of the sp
ace used in the /var directory falls below 75 percent."
        SelectionString  = "Name == \"/var\""
        Severity         = "i"
        NodeNames        = {}
        MgtScope         = "l"
        Toggle           = "Yes"
```

This **lscondition** output (described in detail in Table Table 49 on page 155 in "Creating, modifying and removing conditions" on page 154) shows that the predefined condition `"/var space used"` is very similar to what you are looking for; the only difference is that it triggers an event when the **/var** file system is 90 percent full instead of 85 percent full. While you could just modify the `"/var space used"` condition itself (as described in "Modifying a condition" on page 164), you think it's best to leave this predefined condition as it is, and instead copy it to a new condition. The following **mkcondition** command creates a condition named `"/var space 85% used"` that copies the `"/var space used"` condition, modifying its event expression.

```
mkcondition -c "/var space used" -e "PercentTotUsed > 85" -d "An event
will be generated when more than 85 percent" "/var space 85% used"
```

In the preceding command:

- `-c "/var space used"` indicates that you want to use the `"/var space used"` condition as a template for the new condition.
- `-e "PercentTotUsed > 85"` modifies the condition's event expression.
- `-d "An event will be generated when more than 85 percent"` modifies the condition's event description to reflect the new event expression.
- `"/var space 85% used"` is the name for the new condition.

After running the above command, the `"/var space 85% used"` condition will be included in the list generated by the **lscondition** command, showing that the condition is available for use in a condition/response associated. To see the new condition's detailed information, enter:

```
lscondition "/var space 85% used"
```

Which will display the following output:

```
Displaying condition information:

condition 1:
        Name             = "/var space 85% used"
        Node             = "c175n06.ppd.pok.ibm.com"
        MonitorStatus    = "Not monitored"
        ResourceClass    = "IBM.FileSystem"
        EventExpression  = "PercentTotUsed > 85"
        EventDescription = "An event will be generated when more than 85 percent"
        RearmExpression  = "PercentTotUsed < 75"
        RearmDescription = "The event will be rearmed when the percent of the spa
ce used in the /var directory falls below 75 percent."
        SelectionString  = "Name == \"/var\""
        Severity         = "i"
        NodeNames        = {}
        MgtScope         = "l"
        Toggle           = "Yes"
```

Notice that the new condition is an exact copy of the `"/var space used"` condition except for the modifications specified on the **mkcondition** command.

If you want to prevent user modification or removal of this condition, you could lock it. For more information, see “Locking and unlocking conditions, compound conditions, responses, and condition/response links” on page 194.

If the condition you want to copy is defined on another node of a peer domain or management domain, you can specify the node name along with the condition. For example:

```
mkcondition -c "/var space used":nodeA -e "PercentTotUsed > 85" -d "An event
will be generated when more than 85 percent" "/var space 85% used"
```

Note: Specifying the management server name as the target node is not valid if you are running the command on the management server.

If you are working on the management server, the conditions will be created on the management server without specifying a target. Do not specify the management server name as the target node in any of the following commands:

- **lscondition**
- **lsresponse**
- **lscondresp**
- **mkcondresp**
- **startcondresp**
- **stopcondresp**
- **rmcondresp**

- **rmcondition**
- **chresponse**

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). For more information, see the **mkcondition** command man page and "Target nodes for a command" on page 125.

This next example illustrates two other flags of the **mkcondition** command. The **-E** flag specifies a rearm expression, and the **-D** flag modifies the rearm expression description.

```
mkcondition -c "/var space used" -E "PercentTotUsed < 70" -D "The event will be
rearmed when the percent of the space used in the /var directory falls below 70
percent." "modified /var space used"
```

This next example illustrates the flags of the **mkcondition** command that you can use to set the condition's monitoring scope. The condition's monitoring scope refers to the node or set of nodes where the condition is monitored. Although a condition resource is defined on a single node, its monitoring scope could be the local node only, all the nodes of a peer domain, select nodes of a peer domain, all the nodes of a management domain, or select nodes of a management domain. If the monitoring scope indicates nodes of a peer domain or management domain, the node on which the condition resource is defined must be part of the domain. The monitoring scope is, by default, the local node on which the condition resource resides. To specify a peer domain or management domain, you use the **-m** option. The setting **-m p** indicates a peer domain monitoring scope, and **-m m** indicates a management domain monitoring scope. (The **-m m** option is allowed only if you are defining the condition on the management server of the management domain.) To further refine this monitoring scope, you can use the **-n** option to specify select nodes in the domain. In this next example, we copy the "`/var space used`" condition, but modify its monitoring scope to certain nodes in a peer domain.

```
mkcondition -c "/var space used" -m p -n nodeA,nodeB "/var space used nodeA,nodeB"
```

Finally, suppose that you want a condition that generates an event when the **/usr** file system is 90 percent full. You could again copy the "`var space used`" condition, this time using the **mkcondition** command's **-s** option to specify a different selection string expression. (Since the rearm expression description mentions the **/var** file system, we will modify that as well.)

```
mkcondition -c "/var space used" -s "Name == \"/usr\"" -D "The event will
be rearmed when the percent of the space used in the /usr directory falls
below 75 percent." "/usr space used"
```

In the above example, modifying the event expression was fairly straightforward. Expressions in RMC are discussed in more detail in "Using expressions to specify condition events and command selection strings" on page 196. Here it suffices to say that the syntax of the selection string is similar to an expression in the C programming language or the *where* clause in SQL. In this case, the condition uses the expression "`Name == \"/usr\"`", so that the condition applies only to resources in the class whose `Name` persistent attribute value is `/usr`.

Creating a condition from scratch:

The predefined conditions will usually meet your monitoring needs, at times requiring only minor modifications. However, if no existing condition is similar to the one you want to create, it may be necessary to define the condition completely.

Before you define a condition, familiarize yourself with the basic attributes of a condition. See the table entitledTable 49 on page 155 in "Creating, modifying and removing conditions" on page 154 which describes the attributes of a condition using the predefined condition `/var space used` as an example.

Once you understand the attributes of a condition, use the following steps to create a condition. You will need to provide a significant amount of information to the **mkcondition** command when defining a condition from scratch. The steps that follow are ordered to assist your careful consideration of the purpose and implications associated with each piece of information required.

Complete the following steps to create a condition:

1. **Identify the attribute you want to monitor.** While resource classes define both persistent and dynamic attributes, it is usually dynamic attributes that are monitored. This is because a persistent attribute is less likely to change (and then only by someone explicitly resetting it). An instance of the **Processor** resource class, for example, has a persistent attribute **ProcessorType** that identifies the type of processor. It would be pointless to monitor this attribute; it's not going to change. Dynamic attributes, however, track changing states. An instance of the **Processor** resource class, for example, has a dynamic attribute **OpState** that indicates whether the operational state of the processor is online or offline.

   For monitoring data, the key resource managers are the host resource manager and the file system resource manager. These two resource managers contain the resource classes whose dynamic attributes reflect variables to monitor.

   - The host resource manager enables you monitor system resources for individual machines. In particular, it enables you to monitor operating system load and status.
   - The file system resource manager enables you to monitor file systems. In particular, it enables you to monitor the percentage of disk space and the percentage of i-nodes used by individual file systems.

   For more information on the resource managers and the resource classes they define, see "Resource managers provided with RSCT" on page 110.

   If you have additional resource managers provided by other programs, such such as Tivoli System Automation, see that program's documentation for information about additional resource classes and which attributes they enable you to monitor. You can also examine the available resource classes and attributes using RMC commands (such as the **lsrsrc** command). See "How RMC and the resource managers enable you to manage resources" on page 119 for more information on RMC commands. For complete syntax information on the commands, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

   Note: If, after examining the dynamic attributes provided by the available resource managers, you determine that there are none that contain the value you want to monitor, you can extend RMC by creating a *sensor*. A sensor is a command to be run by RMC (at specified intervals and/or when you explicitly request for it to be run) to retrieve the value you want to monitor. See "Creating event sensor commands for monitoring" on page 177 for more information.

   For example, suppose that you are interested in monitoring the operational state of processors, and would like the system to notify you if a processor goes offline. (There is, in fact, a predefined condition designed to monitor this, but for the sake of this discussion, we'll assume it was accidentally removed.) To see if there are any resource classes that represent processors, you can issue the **lsrsrc** or **lsrsrcdef** command without any parameters or flags.

   ```
   lsrsrc
   ```

   This displays output similar to the following:

   ```
   class_name
   "IBM.Association"
   "IBM.ATMDevice"
   "IBM.AuditLog"
   "IBM.AuditLogTemplate"
   "IBM.Condition"
   "IBM.EthernetDevice"
   "IBM.EventResponse"
   "IBM.FDDIDevice"
   "IBM.Host"
   ```

```
"IBM.FileSystem"
"IBM.PagingDevice"
"IBM.PhysicalVolume"
"IBM.Processor"
"IBM.Program"
"IBM.TokenRingDevice"
⋮
```

The IBM.Processor resource class sounds promising. For details on the resources in this class, enter the following **lsrsrc** command. The **-A d** instructs the command to list only dynamic attributes.

```
lsrsrc -A d IBM.Processor
```

This displays output similar to the following:

```
Resource Dynamic Attributes for: IBM.Processor
resource 1:
        PctTimeUser   = 0.0972310851777207
        PctTimeKernel = 0.446023453293117
        PctTimeWait   = 0.295212932824663
        PctTimeIdle   = 99.1615325287045
        OpState       = 1
resource 2:
        PctTimeUser   = 0.0961145070660594
        PctTimeKernel = 0.456290452125732
        PctTimeWait   = 0.30135492264433
        PctTimeIdle   = 99.1462401181639
        OpState       = 1
resource 3:
        PctTimeUser   = 0.102295524109806
        PctTimeKernel = 0.475051721639257
        PctTimeWait   = 0.316998288621668
        PctTimeIdle   = 99.1056544656293
        OpState       = 1
resource 4:
        PctTimeUser   = 0.0958503317766613
        PctTimeKernel = 0.452945804277402
        PctTimeWait   = 0.30571948042647
        PctTimeIdle   = 99.1454843835195
        OpState       = 1
```

The preceding output shows us that there are five dynamic attributes. While you can get detailed information about these attributes using the **lsrsrcdef** command (as described in "Displaying attribute definition information for a resource or a resource class" on page 135), the names are fairly self-explanatory. The `OpState` attribute monitors whether the processor is online or offline, while the others represent the percentage of time the processor spends in various states. (Of course, the Host resource manager provides predefined conditions for all of these dynamic attributes, so you would not have to create a condition from scratch and could instead either use the predefined conditions as is, or follow the instructions in "Creating a condition by copying an existing one" on page 158. For the sake of this discussion, we'll assume no predefined conditions are available.)

Now that we've found a dynamic attribute (`OpState`) that contains the information we want to monitor, we can move on the the next step.

2. **Design an event expression that will test the attribute for the condition of interest.** Once you have identified the attribute that contains the information you want to monitor, design the event expression you will supply to the **mkcondition** command. An event expression includes the attribute, a mathematical comparison symbol, and a constant. RMC will periodically poll the resource class to determine if this expression is true. If the expression does test true, RMC will execute any response scripts associated with the condition's linked responses.

   RMC keeps track of the previously observed value of an attribute. If an event expression appends an attribute name with "@P", this refers to the previously observed value of the attribute. An event expression might use this capability to compare the currently observed value of the attribute with its

previously-observed value. For example, the following event expression, if specified on a condition, would trigger an event if the average number of processes on the run queue has increased by 50% or more between observations:

```
(ProcRunQueue - ProcRunQueue@P) >= (ProcRunQueue@P * 0.5)
```

Expressions in RMC are described in more detail in "Using expressions to specify condition events and command selection strings" on page 196.

In our example, we want to create a condition that creates an event when a processor goes offline. We've found that the `OpState` dynamic attribute of the Processor resource class contains this information. If the value of `OpState` is 1, the processor is online. The expression `"OpState != 1"` will therefore test true if the processor is offline.

3. **Design a rearm event expression if you determine that one is necessary.** To determine whether a rearm event expression is needed in this condition, consider how the condition will behave later when you have started monitoring it. In our example, RMC will periodically poll the Processor resource class to determine if the expression `"OpState != 1"` tests true. If it does, the event occurs, triggering the condition's linked responses. If there is a rearm expression defined, RMC will, the next time it polls the Processor resource class, test the rearm expression. It will continue to test the rearm expression, until it tests true; only then will RMC resume testing the event expression. If the condition has no rearm expression, then RMC will continue to test the event expression each time it polls the Processor resource class. The linked responses will be triggered each time the event expression is evaluated until the processor is brought back online. Since the linked response might be send e-mail to root or notify everyone on the system, you probably only want this happening once when the processor is first detected offline. We will use `"OpState == 1"` as our rearm expression; the condition will be rearmed only after the processor is detected to be back online.
4. **Determinine the condition's monitoring scope.** If you are on a cluster of nodes configured into management and/or peer domains, the condition's monitoring scope refers to the node or set of nodes where the condition is monitored. Although a condition resource is defined on a single node, its monitoring scope could be the local node only, all the nodes of a peer domain, select nodes of a peer domain, all the nodes of a management domain, or select nodes of a management domain. The monitoring scope is, by default, the local node on which the condition resource resides. To specify a peer domain or management domain, you use the **-m** option. The setting **-m p** indicates a peer domain monitoring scope, and **-m m** indicates a management domain monitoring scope. (The **-m m** option is allowed only if you are defining the condition on the management server of the management domain.) To further refine this monitoring scope, you can use the **-n** option to specify select nodes in the domain.

   Our example plans to monitor the local node on which the condition is defined. Since this is the default behavior, it does not require the **-m** flag.

   For more information on domains in a cluster, see "Management domains and peer domains" on page 3. For more information on the **-m** flag, see the **mkcondition** command's online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.
5. **Design a selection string if you determine that one is necessary.** By default, the condition will apply to all resources in the class. However, a selection string expression, if provided, will filter the resource class so that the condition will apply only to resources that match the expression. The event expression can filter the resource class using any of its persistent attributes. To understand how this works, let's look at the resources in the Processor resource class. The following **lsrsrc** command lists each resource in the Processor resource class. The **-A p** instructs the command to list only the persistent resource attributes of the resources.

   ```
   lsrsrc -A p IBM.Processor
   ```

   The following output is returned.

   ```
   Resource Persistent Attributes for: IBM.Processor
   resource 1:
           Name          = "proc3"
           NodeNameList  = {"c175n06.ppd.pok.ibm.com"}
           ProcessorType = "PowerPC_604"
   ```

```
resource 2:
        Name          = "proc2"
        NodeNameList  = {"c175n06.ppd.pok.ibm.com"}
        ProcessorType = "PowerPC_604"
resource 3:
        Name          = "proc1"
        NodeNameList  = {"c175n06.ppd.pok.ibm.com"}
        ProcessorType = "PowerPC_604"
resource 4:
        Name          = "proc0"
        NodeNameList  = {"c175n06.ppd.pok.ibm.com"}
        ProcessorType = "PowerPC_604"
```

Here we can see that there are four processors that, by default, will all be monitored by the condition. For our example condition, this is the behavior we are looking for. If for some reason we wanted to monitor only the processor named "proc3", we would use the selection string `"Name = "proc3""`.

6. **Determine the severity of the event.** Should the event be considered a critical error, a warning, or merely informational. We'll consider our example condition informational.
7. **Create the condition using the mkcondition command.** Now it's time to put it all together. The following **mkcondition** command defines our condition.

   ```
   mkcondition -r IBM.Processor -e "OpState != 1" -d "processor down"
   -E "OpState == 1" -D "processor online" -S i "new condition"
   ```

   In the preceding command:
   - the **-r** flag specifies the resource class containing the attribute to be monitored.
   - the **-e** flag specifies the event expression.
   - the **-d** flag specifies a short description of the event expression.
   - the **-E** flag specifies the rearm expression.
   - the **-D** flag specifies a short description of the event expression.
   - the **-S** flag specifies the severity of the condition.

If you wanted to prevent user modification or removal of this condition, you could lock it. For more information, see"Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

For detailed syntax information on the **mkcondition** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying a condition:

To modify a condition, you use the **chcondition** command.

Because the **chcondition** command uses the same flags as the **mkcondition** command, you need only supply the **chcondition** command with the name of the condition to change and any changes you want to make.

Examples:

- To modify the event expression and event description of the `"/var space used"` condition, you would use the **-e** and **-d** flags:

  ```
  chcondition  -e "PercentTotUsed > 85" -d "An event
  will be generated when more than 85 percent" "/var space used"
  ```

- To modify the rearm event expression and rearm description, you would use the **-E** and **-D** flags:

  ```
  chcondition -E "PercentTotUsed < 70" -D "The event will be
  rearmed when the percent of the space used in the /var directory falls below 70
  percent." "/var space used"
  ```

- To modify the condition's selection string expression, you would use the **-s** flag:
  ```
  chcondition -s "Name == \"/usr\"" "/var space used"
  ```
- To rename a condition, you would use the **-c** flag. For instance, the condition in the preceding examples should probably not be called "/var space used" anymore, since the selection string has been modified so that the condition applies to the **/usr** file system. To change the name of this condition from "/var space used" to "/usr space used", you would enter:
  ```
  chcondition -c "/usr space used" "/var space used"
  ```

You will not be able to modify a condition that is locked. Instead, the **chcondition** command will generate an error informing you that the condition is locked. For more information on unlocking a condition so it can be modified, see "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

For detailed syntax information on the **chcondition** command, see its online man page.

Removing a condition:

The **rmcondition** command enables you to remove a condition.

For example:
```
rmcondition "/usr space used"
```

If the condition you have specified has linked responses, an error message will display and the condition will not be removed. To remove a condition even if it has linked responses, use the **-f** (force) flag.

Example: The following command removes the "/usr space used" condition even if it has linked responses:
```
rmcondition -f "/usr space used"
```

If the condition is used in a compound condition, the condition cannot be removed even if the **-f** flag is specified. The condition must first be removed from the compound condition, either by using the **rmccondition** command to remove the compound condition or the **chccondition** command to remove the condition from the compound condition.

If the condition you want to remove is defined on another node of a peer domain or management domain, you can specify the node name along with the condition.

Example: The following command removes the "/usr space used" condition that is defined on node *nodeA*:
```
rmcondition "/usr space used":nodeA
```

Note: Specifying the management server name as the target node is not valid if you are running the command on the management server.

If you are working on the management server, the conditions will be created on the management server without specifying a target. Do not specify the management server name as the target node in any of the following commands:
- **lscondition**
- **lsresponse**
- **lscondresp**
- **mkcondresp**
- **startcondresp**
- **stopcondresp**

- **rmcondresp**
- **rmcondition**
- **chresponse**

You will not be able to remove a condition that is locked. Instead, the **rmcondition** command will generate an error informing you that the condition is locked. For more information on unlocking a condition so it can be removed, see "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). For more information, see the **rmcondition** command man page and "Target nodes for a command" on page 125.

For detailed syntax information on the **rmcondition** command, see its online man page.

Creating, modifying, and removing responses:

There are three commands that you can use to manipulate responses.

You can:

- Create a new response using the **mkresponse** command.
- Modify a response using the **chresponse** command.
- Remove a response using the **rmresponse** command.

Before we discuss these commands, it is important that you understand the basic attributes of a response. In "Listing responses" on page 138, we discuss the **lsresponse** command that enables you to list responses that are available. This command lists the predefined responses we provide, as well as any you define. Specifying the name of a response as a parameter to the **lsresponse** command returns detailed information about the response. For example, entering this command:

```
# lsresponse "Informational notifications"
```

Returns the following information about the predefined response `"Informational notifications"`.

```
Displaying response information:

        ResponseName    = "Informational notifications"
        Node            = "c175n06.ppd.pok.ibm.com"
        Action          = "Log info event"
        DaysOfWeek      = 1-7
        TimeOfDay       = 0000-2400
        ActionScript    = "/usr/sbin/rsct/bin/logevent /tmp/infoEvents"
        ReturnCode      = -1
        CheckReturnCode = "n"
        EventType       = "b"
        StandardOut     = "n"
        EnvironmentVars = ""
        UndefRes        = "n"

        ResponseName    = "Informational notifications"
        Node            = "c175n06.ppd.pok.ibm.com"
        Action          = "E-mail root"
        DaysOfWeek      = 2-6
        TimeOfDay       = 0800-1700
        ActionScript    = "/usr/sbin/rsct/bin/notifyevent root"
        ReturnCode      = -1
        CheckReturnCode = "n"
```

```
EventType       = "b"
StandardOut     = "n"
EnvironmentVars = ""
UndefRes        = "n"
```

Each block of information in the preceding output represents a different action associated with the response. You can think of a response as a wrapper around the actions that can be performed when any condition linked with the response tests true. When such a condition event occurs, the response is triggered, and any number of its actions may then be executed. When adding an action to a response, you specify the day(s) of the week and hour(s) of the day when the response action can execute. If the linked condition event occurs during a time when the response action is defined to run, it will execute. Otherwise, the response action will not execute. This enables the system to respond one way to an event during work hours, and another way outside work hours. The preceding command output, for example, shows that during work hours, the response action will be to e-mail root. Outside work hours, however, the response action is to merely log the information.

It is important to understand the information contained in the preceding output, because you can set many of these values using the various flags of the **mkresponse** and **chresponse** commands. Table 51 describes each line of **lsresponse** command output in more detail.

Table 51. Explanation of **lsresponse** *command output*

This line of the lsresponse command output...	Indicates...	Notes
`ResponseName = "Informational notifications"`	The name of the response. In this case `"Informational notifications"`.	Specified as a parameter of the **mkresponse** and **chresponse** commands.
`Node = "c175n06.ppd.pok.ibm.com"`	The node on which the response is defined. This is important, because, when you create a condition/response association, both the condition and the response must reside on the same node. In this case, the `"E-mail root off-shift"` response is defined on the node `"c175n06.ppd.pok.ibm.com"`. This node information is provided only if the management scope is a peer domain scope or a management domain scope.	By default, will be the node where the **mkresponse** command runs. Can be explicitly specified using the **mkresponse** command's **-p** flag.
`Action = "E-mail root"`	The name of this response action. This name describes what the action script does.	Specified by the **-n** flag of both the **mkresponse** and **chresponse** commands.
`DaysOfWeek = 2-6`	The days of the week that this response action can execute. The days of the week are numbered from 1 (Sunday) to 7 (Saturday). This particular response action will not execute on weekends. If the response is triggered on Saturday or Sunday, this response action will not run.	Specified by the **-d** flag of both the **mkresponse** and **chresponse** commands.
`TimeOfDay = 0800-1700`	The range of time during which the response action can execute. This particular response action will execute only during work hours (between 8 am and 5 pm). If the response is triggered outside of these hours, this response action will not run.	Specified by the **-t** flag of both the **mkresponse** and **chresponse** commands.
`ActionScript = "/usr/sbin/rsct/bin/notifyevent root"`	The full path to the script or command to run for this response action. This particular script will e-mail the event information to root.	Specified by the **-s** flag of both the **mkresponse** and **chresponse** commands.
`ReturnCode = -1`	The expected return code of the action script.	Specified by the **-r** flag of both the **mkresponse** and **chresponse** commands.

Table 51. Explanation of **lsresponse** *command output (continued)*

This line of the lsresponse command output...	Indicates...	Notes
`CheckReturnCode = "n"`	Whether or not RMC compares the action script's actual return code to its expected return code. If RMC does make this comparison, it will write a message to the audit log indicating whether they match. If RMC does not make this comparison, it will merely write the actual return code to the audit log. For more information on the the audit log, see "Using the audit log to track monitoring activity" on page 145.	Implied by specifying an expected return code using the **-r** flag of both the **mkresponse** and **chresponse** commands.
`EventType = "b"`	Whether this response action should be triggered for the condition's event, rearm event, or both the event and rearm event. This response action applies to both the event and rearm event. If either the event expression or the rearm expression of a condition linked to this response tests true, this action can be triggered.	Specified by the **-e** flag of both the **mkresponse** and **chresponse** commands.
`StandardOut = "n"`	Whether standard output should be directed to the audit log. For more information on the audit log, see "Using the audit log to track monitoring activity" on page 145.	Specified by the **-o** flag of both the **mkresponse** and **chresponse** commands.
`EnvironmentVars = ""`	Environment variables that RMC should set prior to executing the action script. This enables you to create general-purpose action scripts that respond differently, or provide different information, depending on the environment variable settings. (In addition to any environment variables you define this way, also be aware that RMC sets many variables that the action script can use. For more information, see Table 53 on page 172 in "Creating new response scripts" on page 171.)	Specified by the **-E** flag of both the **mkresponse** and **chresponse** commands.
`UndefRes = "n"`	Indicates whether or not RMC should still execute the action script if the resource monitored by the condition becomes undefined.	Specified by the **-u** flag of both the **mkresponse** and **chresponse** commands.

The **mkresponse** command creates the response with, optionally, one action specification. To add additional actions to the response, you can then use the **chresponse** command. The **chresponse** command also enables you to remove an action from the response, or rename the response. The **rmresponse** command removes a response when it is no longer needed.

In addition to any responses you create, be aware that we provide predefined responses. These are described in Table Table 41 on page 123 in "Predefined and user-defined responses" on page 123.

Creating a response:

Use the **mkresponse** command to create a response. Before creating a response, however, determine whether any of the predefined responses that RSCT provides are suitable for your purposes.

See the table entitledTable 41 on page 123 in "Predefined and user-defined responses" on page 123. For instructions on listing the predefined responses available on your system, see "Listing responses" on page 138. If you find a predefined response that meets your need, you do not need to preform the advanced task of creating a response; instead, you can proceed to "Creating a condition/response association" on page 140 and "Starting condition monitoring" on page 142.

Once you understand the information contained in Table 51 on page 167 in "Creating, modifying, and removing responses" on page 166, you can use the following steps to create a response. Keep in mind that the **mkresponse** command enables you to define one response action only. In fact, with the exception of the response name, the information you supply to this command describes the action. Once you have defined the response using the **mkresponse** command, you can add more actions to it using the **chresponse** command.

Do the following to create a response:

1. **Decide which action script, if any, should be triggered by the response.** There are a number of predefined action scripts that you can associate with the response action. You can also create you own action script and associate it with the response action. In addition, information about the response occurring will be entered into the audit log. You do not need to associate an action script with the action; if you do not, the response information will still be entered into the audit log.

   Table 52 describes the predefined action scripts, which are located in the directory **/usr/sbin/rsct/bin/**.

Table 52. Predefined response scripts provided by RSCT

Response script	Description
displayevent	Available on Linux nodes only. Sends a message about the event to a specified X-window display.
logevent	Logs information about the event to a specified log file. The name of the log file is passed as a parameter to the script. This log file is not the audit log; it is a file you specify.
msgevent	Available on Linux nodes only. Sends information about the event to a specified user's console.
notifyevent	E-mails information about the event to a specified user ID. This user ID can be passed as a parameter to the script, or else is the user who ran the command.
snmpevent	Sends a Simple Network Management Protocol (SNMP) trap to a host running an SNMP event.
wallevent	Broadcasts the event information to all users who are logged in.

   Note: The **/usr/sbin/rsct/bin/** directory also contains variations of three of these scripts called **elogevent**, **enotifyevent**, and **ewallevent**. These have the same functionality as the scripts outlined in the preceding table; the only difference is that they always return messages in English, while the scripts outlined in Table 52 return messages based on the local language setting.

   In addition to our predefined scripts which, as you can see from the preceding table, perform general-purpose actions, you can also create your own action scripts. One reason you might do this is to create a more targeted response to an event. For example, you might want to write a script that would automatically delete the oldest unnecessary files when the **/tmp** file system is 90 percent full. For more information, see “Creating new response scripts” on page 171.

   If you decide to use one of our predefined action scripts, be sure that you understand exactly what the script will do. For more information on a script, see the script's online man page.

   Whether you choose one of our predefined scripts or one you create, you will specify it to using the **mkresponse** command's **-s** flag. You will need to provide the full path name of the script and any parameters that you need or want to pass to it.

   Example: Suppose that you want to use the log event script to log the event information to the file **/tmp/EventLog**. The specification would be:

   ```
   -s "/usr/sbin/rsct/bin/logevent /tmp/EventLog"
   ```

2. **Decide on the days/hours during which this response action can be run.** Some response actions may only be appropriate or desired during work hours, some may only be desired outside work hours. Often a response will have multiple actions, each designed for different days or times. For example, one response action might be defined to run only during work hours and would notify you by e-mail about an error. Another action on the same response might run only outside work hours and would merely log the error to a file.

   The **mkresponse** command's **-d** option specifies the days of the week that the command can execute. The days are numbered from 1 (Sunday) to 7 (Saturday). You can specify either a single day (7), multiple days separated by a plus sign (1+7), or a range or days separated by a hyphen (2-6).

   Using the **mkresponse** command's **-t** flag, you can specify the range of time during which the command can run. The time is specified in a 24-hour format, where the first two digits represent the hour and the second two digits are the minutes. The start time and end time are separated by a hyphen.

   Example: If you want the response action to run only during work hours (Monday through Friday, 8 AM to 5 PM), the specification would be:

```
-d 2-6 -t 0800-1700
```

You can also specify different times for different days by making multiple specifications with the **-d** and **-t** flags. The number of day parameters must match the number of time parameters.

Example: If you want the response action to be used anytime Saturday and Sunday, but only between 8 a.m. and 5 p.m. on weekdays, you would use the following specification:

```
-d 1+7,2-6 -t 0000-2400,0800-1700
```

3. **Decide if this response action should apply to the condition event, condition rearm event, or both.** You specify this using the **-e** flag with the setting **a** (event only), **r** (rearm event only), or **b** (both event and rearm event).

   Example: If you want the action to be executed in response to the condition event only, the specification would be:

   ```
   -e a
   ```

4. **Create the response using the mkresponse command.** Once you understand the response action you want to define, you can enter the **mkresponse** command with all the appropriate option settings. Use the **-n** flag to specify the action name, and pass the response name as a parameter to the command.

   Example:

   ```
   mkresponse -n LogAction -s /usr/sbin/rsct/bin/logevent /tmp/EventLog
   -d 1+7,2-6 -t 0000-2400,0800-1700 -e a "log info to /tmp/EventLog"
   ```

   The preceding command creates a response named "log info to /tmp/EventLog". If you wanted to prevent user modification or removal of this response, you could lock it. For more information, see “Locking and unlocking conditions, compound conditions, responses, and condition/response links” on page 194.

To add additional actions to a response, use the **chresponse** command, as described in “Modifying a response” on page 175.

Targeting nodes:

The **mkresponse** command is affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope. The **mkresponse** command's **-p** flag, if specified, indicates the name of a node where the response is defined. This must be a node within the management scope for the local node (or the node indicated by the CT_CONTACT environment variable).

If the CT_MANAGEMENT_SCOPE environment variable is not set, and the **-p** flag is used, this command will attempt to set the CT_MANAGEMENT_SCOPE environment variable to the management scope that contains the node specified on the **-p** flag. In this case, the specified node should be in the management domain or peer domain of the local node (or the node indicated by the CT_CONTACT environment variable).

If using the **mkresponse** command on a management server, do not specify the **-p** flag if you want the condition to be defined on the management server.

For more information, see the **mkresponse** command man page and “Target nodes for a command” on page 125.

For detailed syntax information on the **mkresponse** command, see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Creating new response scripts:

The predefined response scripts we provide are general purpose ways of notifying users about an event, or else logging the event information to a file. In addition to these general-purpose scripts, you might want to write you own scripts that provide more specific responses to events.

You might want to do this to create an automatic recovery script that would enable RMC to solve a simple problem automatically. For example when the **/tmp** directory is over 90 percent full, you could have RMC run a script to automatically delete the oldest unnecessary files in the **/tmp** directory. Another reason you might want to create your own scripts is to tailor system responses to better suit your particular organization. For example, you might want to create a script that calls your pager when a particular event occurs.

If you want to create your own response scripts, it pays to examine the existing scripts that RSCT provides (as described in Table Table 52 on page 169 in "Creating a response" on page 168). These scripts are located in the directory **/usr/bin/rsct/bin**, and can be useful as templates in creating your new scripts, and also illustrate how the script can use ERRM environment variables to obtain information about the event that triggered its execution. For example, suppose that you wanted to create a script that called your pager when particular events occur. You might want to use our predefined script **wallevent** as a template in creating your new script. This predefined script uses the **wall** command to write a message to all users who are logged in. You could make a copy of this program, and replace the **wall** command with a program to contact your pager.

Note: Because our predefined responses use the predefined response scripts, do not modify the original scripts in **/usr/bin/rsct/bin**. In you want to use an existing script as a template for a new script, copy the file to a new name before making your modifications.

After a condition event occurs, but before the response script executes, ERRM sets a number of environment variables that contain information about the event. The script can check the values of these variables in order to provide the event information to the user. Using the ERRM environment variables, the script can ascertain such information whether it was triggered by the condition event or rearm event, the time the event occurred, the host on which the event occurred, and so on.

Example: The following is a predefined Perl script called **wallevent** which illustrates the use of the ERRM environment variables. The ERRM environment variables names begin with "**ERRM_**" and are highlighted in the example.

```
# main()

PERL=/usr/sbin/rsct/perl5/bin/perl

CTMSG=/usr/sbin/rsct/bin/ctdspmsg
MSGMAPPATH=/usr/sbin/rsct/msgmaps
export MSGMAPPATH

Usage=`$CTMSG script IBM.ERrm.cat MSG_SH_USAGE`

while getopts ":h" opt
do
  case $opt in

    h ) print "Usage: `basename $0` [-h] "
        exit 0;;

    ? ) print "Usage: `basename $0` [-h] "
        exit 3;;
  esac
done

# convert time string
seconds=${ERRM_TIME%,*}
```

```
EventTime=$(seconds=$seconds $PERL -e \
'
use POSIX qw(strftime);
print strftime("

'
)

WallMsg=`$CTMSG script IBM.ERrm.cat MSG_SH_WALLN "$ERRM_COND_SEVERITY"
"$ERRM_TYPE" "$ERRM_COND_NAME" "$ERRM_RSRC_NAME"
"$ERRM_RSRC_CLASS_NAME" "$EventTime" "$ERRM_NODE_NAME"
"$ERRM_NODE_NAMELIST"`

wall "${WallMsg}"


#wall "$ERRM_COND_SEVERITY $ERRM_TYPE occurred for the condition $ERRM_COND_NAME
on the resource $ERRM_RSRC_NAME of the resource class $ERRM_RSRC_CLASS_NAME at
$EventTime on $ERRM_NODE_NAME"
```

The preceding script uses the **ERRM_TIME** environment variable to ascertain the time that the event occurred, the **ERRM_COND_SEVERITY** environment variable to learn the severity of the event, the **ERRM_TYPE** environment variable to determine if it was the condition event or rearm event that triggered the script's execution, and so on. This information is all included in the message sent to online users.

Table 53 describes the ERRM environment variables that you can use in response scripts. Unless otherwise specified, these environment variables are available for ERRM commands in non-batched event responses, batched event responses, and batched event files.

Table 53. ERRM environment variables

This environment variable...	Contains...
ERRM_ATTR_NAME **ERRM_ATTR_NAME_***n*	The display name of the dynamic attribute used in the expression that caused this event to occur. This environment variable is repeated if the value of **ERRM_ATTR_NUM** is greater than 1. The value of *n* is from 2 to **ERRM_ATTR_NUM**. This environment variable is not available for ERRM commands in batched event responses.
ERRM_ATTR_NUM	The number of attributes that are used in the event expression. This environment variable is not available for ERRM commands in batched event responses.
ERRM_ATTR_PNAME **ERRM_ATTR_PNAME_***n*	The programmatic name of the attribute used in the expression that caused this event to occur. This environment variable is repeated if the value of **ERRM_ATTR_NUM** is greater than 1. The value of *n* is from 2 to **ERRM_ATTR_NUM**. This environment variable is not available for ERRM commands in batched event responses.
ERRM_BATCH_REASON	The reason why the batched event was triggered. The valid values are: **1** (the event batch interval expired), **2** (the maximum number of batching events was reached), **3** (monitoring stopped), **4** (the association between the condition and event response was removed), **5** (the event response was deleted), and **6** (the condition was deleted). This environment variable is not available for ERRM commands in non-batched event responses or batched event files.
ERRM_COND_BATCH	The indication of whether the condition is batching events. The valid values are: **0** (no) and **1** (yes).
ERRM_COND_BATCH_NUM	The number of events in the batched event file. This environment variable is not available for ERRM commands in non-batched event responses or batched event files.

Table 53. ERRM environment variables (continued)

This environment variable...	Contains...
ERRM_COND_HANDLE	The resource handle of the condition that caused the event. The format of this value is six hexadecimal integers that are separated by spaces and written as a string, for example: `0x6004 0xffff 0x180031ae 0xe300b8db 0x10f4de7b 0x40a5c5c9`
ERRM_COND_MAX_BATCH	The maximum number of events that can be batched together, if the condition is batching events. If the value is **0**, there is no limit. This environment variable is not available for ERRM commands in non-batched event responses.
ERRM_COND_NAME	The name of the condition that caused the event.
ERRM_COND_SEVERITY	The severity of the condition that caused the event. For the severity attribute values of 0, 1, and 2, this environment variable has the following values, respectively: **Informational**, **Warning**, and **Critical**. All other severity attribute values are represented in this environment variable as a decimal string. This environment variable is not available for ERRM commands in batched event responses.
ERRM_COND_SEVERITYID	The severity value of the condition that caused the event. The valid valuea are: **0** (Informational), **1** (Warning), and **2** (Critical). This environment variable is not available for ERRM commands in batched event responses.
ERRM_DATA_TYPE **ERRM_DATA_TYPE_***n*	The RMC **ct_data_type_t** type of the attribute that changed to cause this event. The valid values are: **CT_BINARY_PTR**, **CT_CHAR_PTR**, **CT_FLOAT32**, **CT_FLOAT64**, **CT_INT32**, **CT_INT64**, **CT_SD_PTR**, **CT_UINT32**, and **CT_UINT64**. The actual value of the attribute is stored in the **ERRM_VALUE** environment variable (except for attributes with a data type of **CT_NONE**). This environment variable is repeated if the value of **ERRM_ATTR_NUM** is greater than 1. The value of *n* is from 2 to **ERRM_ATTR_NUM**. This environment variable is not available for ERRM commands in batched event responses.
ERRM_ER_HANDLE	The event response resource handle for this event. The format of this value is six hexadecimal integers that are separated by spaces and written as a string, for example: `0x6006 0xffff 0x180031ae 0xe300b8db 0x10ffa192 0xdd39d86b` This environment variable is not available for ERRM commands in batched event files.
ERRM_ER_NAME	The name of the event that triggered this event response script. This environment variable is not available for ERRM commands in batched event files.
ERRM_ERROR_MSG	The descriptive message for **ERRM_ERROR_NUM**, if the value of **ERRM_ERROR_NUM** is not **0**. This environment variable is not available for ERRM commands in batched event responses.
ERRM_ERROR_NUM	The error code from the RMC subsystem for an error event. If the value is **0**, an error did not occur. This environment variable is not available for ERRM commands in batched event responses.
ERRM_EVENT_DETAIL_FILE	The file name of where the batched events can be found, if the condition is batching events. This environment variable does not appear in the batched event file. This environment variable is not available for ERRM commands in non-batched event responses or batched event files.
ERRM_EXPR	The condition event expression or rearm event expression that tested True, thus triggering this linked response. The type of event that triggered the linked response is stored in the **ERRM_TYPE** environment variable. This environment variable is not available for ERRM commands in batched event responses.
ERRM_NODE_NAME	The host name on which this event or rearm event occurred. This environment variable is not available for ERRM commands in batched event responses.

Table 53. ERRM environment variables (continued)

This environment variable...	Contains...
ERRM_NODE_NAMELIST	A list of host names. These are the hosts on which the monitored resource resided when the event occurred. This environment variable is not available for ERRM commands in batched event responses.
ERRM_RSRC_CLASS_NAME	The display name of the resource class containing the attribute that changed, thus causing the event to occur. This environment variable is not available for ERRM commands in batched event responses.
ERRM_RSRC_CLASS_PNAME	The programmatic name of the resource class containing the attribute that changed, thus causing the event to occur. This environment variable is not available for ERRM commands in batched event responses.
ERRM_RSRC_HANDLE	The resource handle of the resource with the state change that caused this event to be generated. The format of this value is six hexadecimal integers that are separated by spaces and written as a string, for example: `0x6009 0xffff 0x180031ae 0xe300b8db 0x10bee2df 0x33b20837` This environment variable is not available for ERRM commands in batched event responses.
ERRM_RSRC_NAME	The name of the resource whose attribute changed, thus causing this event. This environment variable is not available for ERRM commands in batched event responses.
ERRM_RSRC_TYPE	The type of resource that caused the event to occur. The valid values are: **0** (an existing resource), **1** (a new resource), and **2** (a deleted resource). This environment variable is not available for ERRM commands in batched event responses.
ERRM_SD_DATA_TYPES **ERRM_SD_DATA_TYPES_***n*	The data type for each element within the structured data (SD) variable, separated by commas. This environment variable is only defined when **ERRM_DATA_TYPE** is CT_SD_PTR. For example: CT_CHAR_PTR, CT_UINT32_ARRAY, CT_UINT32_ARRAY, CT_UINT32_ARRAY. This environment variable is repeated if the value of **ERRM_ATTR_NUM** is greater than 1. The value of *n* is from 2 to **ERRM_ATTR_NUM**. The **ERRM_SD_DATA_TYPES_***n* environment variable is only defined when the value of **ERRM_DATA_TYPE_***n* is **CT_SD_PTR**. This environment variable is not available for ERRM commands in batched event responses.
ERRM_TIME	The time the event occurred. The time is written as a decimal string representing the time since midnight January 1, 1970 in seconds, followed by a comma and the number of microseconds. This environment variable is not available for ERRM commands in batched event responses.
ERRM_TYPE	The type of event that occurred. For conditions, the valid values are: **Event** and **Rearm Event**. For responses, the valid values are: **Event**, **Rearm Event**, and **Error Event**. This environment variable is not available for ERRM commands in batched event responses.
ERRM_TYPEID	The value of **ERRM_TYPE**. For conditions, the valid values are: **0** (event) and **1** (rearm event). For responses, the valid values are: **0** (event), **1** (rearm event), and **2** (error event). This environment variable is not available for ERRM commands in batched event responses.

Table 53. ERRM environment variables (continued)

This environment variable...	Contains...
ERRM_VALUE **ERRM_VALUE**_*n*	The value of the attribute that caused the event to occur for all attributes except those with a data type of **CT_NONE**. The following data types are represented with this environment variable as a decimal string: CT_INT32, CT_UINT32, CT_INT64, CT_UINT64, CT_FLOAT32, and CT_FLOAT64. CT_CHAR_PTR is represented as a string for this environment variable. CT_BINARY_PTR is represented as a hexadecimal string separated by spaces. CT_SD_PTR is enclosed in square brackets and has individual entries within the SD that are separated by commas. Arrays within an SD are enclosed within braces {}. For example, ["My Resource Name",{1,5,7},{0,9000,20000},{7000,11000,25000}] See the definition of **ERRM_SD_DATA_TYPES** for an explanation of the data types that these values represent. This environment variable is repeated if the value of **ERRM_ATTR_NUM** is greater than 1. The value of *n* is from 2 to **ERRM_ATTR_NUM**. This environment variable is not available for ERRM commands in batched event responses.

Note:

In addition to these ERRM environment variables, you can, when defining a response action using either the **mkresponse** or **chresponse** command, specify additional environment variables for RMC to set prior to triggering the event response script. This enables you to write a more general purpose script that will behave differently based on the environment variables settings associated with the response action. To specify such user-defined environment variables, use the **-E** flag of either the **mkresponse** or **chresponse** command. For example:

```
mkresponse -n "Page Admins"  -s /usr/sbin/rsct/bin/pageevent
-d 1+7 -t 0000-2400  -e a -E 'ENV1="PAGE ALL"' "contact system administrators"
```

Of course, if you do create your own response scripts, test them before use as response actions in a production environment. The **-o** flag of the **mkresponse** and **chresponse** commands is useful when debugging new actions. When specified, all standard output from the script is directed to the audit log. This is useful because, while standard error is always directed to the audit log, standard output is not.

For more information about the predefined response scripts (as well as information on the **-E** and **-o** flags of the **mkresponse** and **chresponse** commands), see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying a response:

To modify a response, you use the **chresponse** command.

You can use this command to:

- add actions to the response
- remove actions from the response
- rename the response

For adding a response action, the **chresponse** command uses the same flags as the **mkresponse** command. You specify the **-a** flag to indicate that you want to add an action, and then define the action using the flags described in "Creating a response" on page 168.

Example: The following command adds an action to a response named `"log info"`:

```
chresponse -a -n LogAction -s /usr/sbin/rsct/bin/logevent /tmp/EventLog
-d 1+7,2-6 -t 0000-2400,0800-1700 -e a "log info"
```

To delete an action from a response specify the **-p** flag on the **chresponse** command. You will also need to specify the response action you want to remove using the **-n** flag.

Example: To remove the response action named "E-mail root" from the response named "E-mail root any time", you would enter the following command:

```
chresponse -p -n "E-mail root" "E-mail root any time"
```

To rename a response, you use the **-c** flag.

Example: To rename the response "E-mail root any time" to "E-mail system administrator", you would enter:

```
chresponse -c "E-mail system administrator" "E-mail root any time"
```

If the response you want to modify is defined on another node of a peer domain or management domain, you can specify the node name along with the response.

Example: The following command modifies the response named "log info" defined on node *nodeA*:

```
chresponse -a -n LogAction -s /usr/sbin/rsct/bin/logevent /tmp/EventLog
-d 1+7,2-6 -t 0000-2400,0800-1700 -e a "log info":nodeA
```

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). For more information, see the **chresponse** command man page and "Target nodes for a command" on page 125.

You will not be able to modify a response that is locked. Instead, the **chresponse** command will generate an error informing you that the response is locked. For more information on unlocking a response so it can be modified, see "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

For detailed syntax information on the **chresponse** command, see its online man page.

Removing a response:

The **rmresponse** command enables you to remove a response.

For example, to remove a response named "E-mail system administrator", enter:

```
rmresponse "E-mail system administrator"
```

If the response you have specified has linked conditions, an error message will display and the response will not be removed. To remove the response even if it has linked conditions, use the **-f** (force) flag.

Example: To remove the response even if it has linked conditions, enter:

```
rmresponse -f "E-mail system administrator"
```

If the response you want to remove is defined on another node of a peer domain or management domain, you can specify the node name along with the response.

Example: To remove the response named "E-mail system administrator" defined on node *nodeA*, enter:

```
rmresponse "E-mail system administrator":nodeA
```

Note: Specifying the management server name as the target node is not valid if you are running the command on the management server.

If you are working on the management server, the conditions will be created on the management server without specifying a target. Do not specify the management server name as the target node in any of the following commands:

- **lscondition**
- **lsresponse**
- **lscondresp**
- **mkcondresp**
- **startcondresp**
- **stopcondresp**
- **rmcondresp**
- **rmcondition**
- **chresponse**

You will not be able to remove a response that is locked. Instead, the **rmresponse** command will generate an error informing you that the response is locked. For more information on unlocking a response so it can be removed, see "Locking and unlocking conditions, compound conditions, responses, and condition/response links" on page 194.

Targeting nodes:

When specifying a node as in the preceding example, the node specified must be a node defined within the management scope (as determined by the CT_MANAGEMENT_SCOPE environment variable) for the local node or the node specified by the CT_CONTACT environment variable (if it is set). For more information, see the **chresponse** command man page and "Target nodes for a command" on page 125.

For detailed syntax information on the **rmresponse** command, see its online man page.

Creating event sensor commands for monitoring:

When none of the attributes of the available resource classes contain the value you are interested in monitoring, you can extend the RMC subsystem by creating a *sensor*. A *sensor* is a command that the RMC subsystem runs to retrieve one or more user-defined values.

You can define a sensor to be run at set intervals, you can run it explicitly, or both. The sensor is essentially a resource that you add to the **IBM.Sensor** resource class of the sensor resource manager. The values returned by the script are dynamic attributes of that resource. You can then create a condition to monitor these dynamic attributes that you have defined.

To create a sensor and to create a condition for monitoring a dynamic attribute that is returned by the sensor follow these steps:

1. **Identify a variable value that none of the existing resource managers currently return.** For example, suppose that you want to monitor the number of users logged on to the system. This is a variable that none of the existing resource managers define. Since there is no existing attribute that contains the value, you will need to create a sensor if you want to monitor this value. Because there is no existing attribute that contains the value, you need to create a sensor if you want to monitor this value.
2. **Create the sensor command script that RMC will run to retrieve the system value(s) of interest.** In our example, we said we wanted to monitor the number of users currently logged on to the system. This following script will retrieve this information:

   ```
   #!/usr/bin/perl
   my @output='who';
   print 'Int32=',scalar(@output), "\n";
   exit;
   ```

   When creating sensor command scripts, be aware of the following:

- The command should return the value it retrieves from the system by sending it to standard output in the form *attribute*=*value*. The *attribute* name used depends on the type of the value and is one of these: **String**, **Int32**, **Uint32**, **Int64**, **Uint64**, **Float32**, **Float64**, or **Quantum**. If only the value is sent to standard output, the attribute name is assumed to be **String**.
- If the command returns more than one type of data, it should send a series of *attribute*=*value* pairs to standard output, separated by blanks (for example: `Int32=10 String="abcdefg"`).

3. **Add your sensor command to the RMC subsystem.** One you have created the sensor command script, you need to add it to the RMC subsystem so that RMC will run the command at intervals to retrieve the value of interest. To do this, you create a sensor object using the **mksensor** command. When entering this command, you need to name the sensor you are creating and provide the full path name of the sensor command script.

   Example: If our sensor command script is **/usr/local/bin/numlogins**, then we could create the sensor named **NumLogins** by entering:

   ```
   mksensor NumLogins /usr/local/bin/numlogins
   ```

   As soon you create the sensor, RMC will periodically run its associated script to retrieve the value. The value will be stored as a dynamic attribute of the Sensor resource. In our example, the number of users currently logged onto the system will be the value of the **NumLogins** resource's **Int32** dynamic attribute.

   By default, RMC will run the sensor command script at 60-second intervals. To specify a different interval, use the **-i** flag of the **mksensor** command.

   Example: To specify that RMC should run the **numlogins** script at five-minute (300-second) intervals, enter:

   ```
   mksensor -i 300 NumLogins /usr/local/bin/numlogins
   ```

   In addition to any interval you set, you can also explicitly run the sensor command using the **refsensor** command.

   Example: The following command runs the **NumLogins** sensor:

   ```
   refsensor NumLogins
   ```

   The **refsensor** command refreshes a sensor and is independent of, and in addition to, the refresh interval you set. If you prefer to only manually run the sensor using the **refsensor** command, you can set the interval to **0**.

   Example: The following command sets the interval to **0**:

   ```
   mksensor -i 0 NumLogins /usr/local/bin/numlogins
   ```

   When creating a sensor, be aware of the following:

   - Because the sensor resource identifies the sensor command script using a full path name. Therefore, the sensor must be defined on the same node as the command script, or otherwise accessible to it (for example, in a shared file system).
   - RMC will run the sensor command script in the process environment of the user who invokes the **mksensor** command. This user should therefore have the permissions necessary to run the command script. If the command script can only be run by the root user, than the root user must issue the **mksensor** command.

4. **Create a condition to monitor a dynamic attribute of the sensor.** The **mksensor** command creates a sensor resource of the **IBM.Sensor** resource class. The sensor command script that is associated with this resource runs at set intervals, when you issue the **refsensor** command, or both. Any value that the script returns is stored as a dynamic attribute of the sensor resource. In the following example, the sensor resource is named **NumLogins**. Because its associated command script contains the statement `print 'Int32=',scalar(@output), "\n";`, the value we're interested in will be available in the **Int32** dynamic attribute. The following condition will trigger an event if any users are logged in to the system:

   ```
   mkcondition -r IBM.Sensor -e "Int32 != 0" -d "users logged in" "users online"
   ```

   You can use the **MonitorStatus** dynamic attribute to find out if certain sensor attributes with values that a sensor command can generate or update are being monitored. These dynamic sensor attributes

are: **String**, **Int32**, **Uint32**, **Int64**, **Uint64**, **Float32**, **Float64**, **Quantum**, and **SD**. A **MonitorStatus** value of **0** indicates that the attribute is not being monitored. A value of **1** means that it is being monitored. The following condition will generate an event whenever the **MonitorStatus** dynamic attribute is changed:

```
mkcondition -r IBM.Sensor -e "MonitorStatus != MonitorStatus@P"
```

You can use the **SD** dynamic attribute to monitor more than one type of data in one condition or to get different types of data in the same event callback. The following condition will generate an event whenever the element **Int32** is updated with a value that is not **0**:

```
mkcondition -r IBM.Sensor -e "SD.Int32 !=0"
```

In addition to being able to create conditions based on the output of the sensor command script, be aware that the exit value of the script is stored in the sensor resource's **ExitValue** attribute, so you can also create a condition based on this. The following condition will generate an event if the exit code is not **0**:

```
mkcondition -r IBM.Sensor -e "ExitValue != 0"
```

ExitValue can be returned only when the exit value from a sensor command is not interpreted as an error that is based on the **ErrorExitValue** persistent attribute.

For more information about the **mksensor** and **refsensor** commands, see their online man pages in the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides. These technical references also contain information about these related commands: **chsensor** (for modifying sensors), **lssensor** (for listing sensor information), **refsensor** (for refreshing sensors) and **rmsensor** (for removing sensors).

Creating event microsensors for monitoring:

Because the **IBM.Sensor** and **IBM.MicroSensor** classes both have a **Name** attribute that is authored by users who are alone aware of any meaning, it is possible that resources of both types could have the same name.

To avoid this conflict, add the **-m** option to a sensor command to distinguish it as the **IBM.MicroSensor** class and not the **IBM.Sensor** class. If the **-m** option is not passed, the default is to operate on the command as belonging to the **IBM.Sensor** class.

Microsensor interface:

All microsensors must export the interface defined by RMC in **ct_microsensor.h**. This C header file provides the data types definitions and the declaration of the routines that all microsensors must export.

All callback functions return a **cu_error_t** structure pointer. They can fill this structure in using the RMC supplied **cu_pkg_error()** function. The microsensor resource manager will release these structs as needed using the RMC **cu_rel_error()** function.

The microsensor resource manager will assume that a NULL **cu_error_t** value pointer value means success, and that a non-NULL **cu_error_t** value pointer value means an error occurred. If an error occurs, most callback functions will assume and render the module unusable. See the *RSCT: RMC Programming Guide* for information about the data types and interface that microsensors must export.

Batching events:

Multiple events can be grouped or "batched" together and passed to a response. The events are grouped according to the time span in which they occur. You can specify a maximum number of events that are to be grouped within the time span.

Grouping a set of events so that a response script can process them together is referred to as *batching*. This batching is done for the events of a condition that occur within a time interval. You can define a condition that batches events, providing a time interval (in seconds) for the batching and an optional maximum number of events that can be in a batch. The batching interval determines when the batch of events is created. When an event for a batching condition is received, a timer is set to the defined batching interval. When that timer expires, the events that have accumulated during the interval are batched together. If a timed interval has no records, batching stops, but will start again for the next event for the condition. If the maximum number of events for a batch is defined, that number of events will be in a batch, at the most. A batch of events is created when the maximum number is reached. The timed interval is reset when the maximum number is reached.

When a monitored condition is batching events, the intermediate events are saved to a file. The batched event file is sent to the responses for processing when the interval timer expires or when the maximum number of events for a batch is reached.

For a monitored condition, you can:

- save files of batched events:
  - up to a maximum total file size. To do this, specify this persistent resource attribute in megabytes: **BatchedEventMaxTotalSize**.
  - until a specified retention period has passed. To do this, specify this persistent resource attribute: **BatchedEventRetentionPeriod**. This attribute indicates how long in hours a batched event file should be kept after all associated response scripts are run.
- control how ERRM writes audit log records. You can use the **AuditLogControl** persistent resource attribute to specify that ERRM writes all audit log records (**0**), only error records (**1**), or no audit log records (**2**).

You can get information about a newly-saved file of batched events by monitoring the **LastBatchedEventFile** dynamic resource attribute. By monitoring this dynamic resource attribute, you can activate a condition without associating it with an event-response.

You can define batching-capable event-responses without actions. You can use this type of event-response to activate a condition on the command line that saves batched event files.

Responses that handle batched events must be defined as supporting batched events. A response can handle either a single event or a file of batched events. The **EventBatching** attribute of the **IBM.EventResponse** class is used to indicate the type of event the response can handle. The response action gets control the way it does for a single event, but the ERRM environment variables are set from the last event in the batch. The **ERRM_EVENT_DETAIL_FILE** environment variable contains the location of the batched events file.

If there are multiple responses for a monitored, batching condition, all responses get the same batched event file. That is, the batching is by condition, not by the condition-response association. If a batched condition is monitored using one response, and this condition becomes associated with another response, the event file and interval timer are shared with the current batch. A second batch interval timer is not created. When a condition-response association becomes inactive, the batched event file is copied and sent to the response. The event batching continues as long as at least one condition-response association is active for a condition.

The batched event files are saved in the **/tmp/errmbatch/** directory. When ERRM starts, it deletes all files in this directory. The individual batched event files are named:

/tmp/errmbatch/*condition_name*.*first_event_date*.*first_event_time*

The batched event file consists of a header stanza followed by event stanzas, one for each event in the batch. The header stanza lists the time of the first and last event and how many events are in the file. Each event stanza lists all of the ERRM envoronment variables and their values for the event, one per

line. The event stanzas are separated by a blank line and contain a header line for each event. In the event stanza, **ERRM_TYPEID** and **ERRM_TYPE** are the first two environment variables listed. Otherwise, there is no particular order of the environment variables within a stanza.

Here is an example of a header and event stanza for a batched event file:

```
FIRST_EVENT_TIME=1224875454,529041
LAST_EVENT_TIME=1224875514,283748
NUM_EVENTS=34

Event 1:
ERRM_TYPEID=0
ERRM_TYPE=Event
ERRM_ATTR_NAME='Percent Total Space Used'
ERRM_ATTR_PNAME=PercentTotUsed
ERRM_COND_HANDLE='0x6004 0xffff 0x180031ae 0xe300b8db 0x10f4de7b 0x40a5c5c9'
ERRM_COND_NAME='/tmp space used'
ERRM_COND_SEVERITY=Informational
ERRM_COND_SEVERITYID=0
ERRM_DATA_TYPE=CT_INT32
ERRM_EXPR='PercentTotUsed>90'
ERRM_NODE_NAME=c175n06.ppd.pok.ibm.com
ERRM_NODE_NAMELIST={c175n06.ppd.pok.ibm.com}
ERRM_RSRC_CLASS_NAME='File System'
ERRM_RSRC_CLASS_PNAME=IBM.FileSystem
ERRM_RSRC_HANDLE='0x6009 0xffff 0x180031ae 0xe300b8db 0x10bee2df 0x33b20837'
ERRM_RSRC_NAME=/tmp
ERRM_RSRC_TYPE=0
ERRM_TIME=1224875454,529041
ERRM_VALUE=92
ERRM_COND_BATCH=1
ERRM_MAX_BATCH=100
.
.
.
Event 2:
ERRM_TYPEID=0
ERRM_TYPE=Event
```

When all of the responses have completed, ERRM deletes the batched event file. If the event data needs to be kept longer, the response script saves a copy of the batched event file.

Caching events:

When an event is monitored, the last event for a condition is saved in the **LastEvent** dynamic attribute of the condition. When caching events is enabled (default), the last event of a condition for each monitored resource is kept in memory by the event response resource manager (ERRM).

Use **runact** command actions to retrieve the cached events. Descriptions of valid class and resource actions follow:

StatLastEvent
: Class action. Displays whether caching is enabled (**Status = 1**) and how many events are currently saved in the cached. It has no parameters.

QueryLastEvent
: Class action. Displays all of the events cached for a particular condition resource since the last time the ERRM subsystem was started. It has no parameters. The output of the action is zero or more structured data values in the form of the **LastEvent** attribute, one for each saved event.

ResetLastEvent
: Class action. Removes events from the caches based on their ages (Age=nn). Saved events older than this age will be removed from the event cache. For example, **Age=0** removes all cached events. **Age=30** removes all saved events older than 30 days from the event cache.

QueryLastEvent

Resource action. Displays all of the cached events for the selection string specified since the last time the ERRM subsystem was started. It has no parameters. The output of the action is 0 or more structured data values in the form of the **LastEvent** attribute, one for each saved event.

Sample output of a **StatLastEvent** class action follows:

```
> runact -c IBM.Condition StatLastEvent
Resource Class Action Response for StatLastEvent
sd_element 1:
         Status    = 1
         NumEvents = 400
```

Sample output of a **QueryLastEvent** class action follows:

```
> runact -c IBM.Condition
QueryLastEvent
Resource Class Action Response for QueryLastEvent
sd_element 1:
         Occurred        = 1
         ErrNum          = 0
         ErrMsg          = ""
         EventFlag       = 6
         EventTimeSec    = 1218641530
         EventTimeUsec   = 507957
         RsrcHndl        = "0x6009 0xffff 0x180031ae
                            0xe300b8db 0x10bee2df 0x33b20837"
         NodeName        = "c175n06.ppd.pok.ibm.com"
         NumAttrs        = 3
         NumAttrsEvtExpr = 1
         AttrArray       = 11
         ACT_RSP_ELE_11  = "PercentTotUsed"
         ACT_RSP_ELE_12  = 2
         ACT_RSP_ELE_13  = 78
         ACT_RSP_ELE_14  = "Name"
         ACT_RSP_ELE_15  = 8
         ACT_RSP_ELE_16  = "/tmp"
         ACT_RSP_ELE_17  = "NodeNameList"
         ACT_RSP_ELE_18  = 19
         ACT_RSP_ELE_19  = {"c175n06.ppd.pok.ibm.com"}
         ACT_RSP_ELE_20  =
         ACT_RSP_ELE_21  =
         ACT_RSP_ELE_22  =
         ACT_RSP_ELE_23  =
         ACT_RSP_ELE_24  =
         ACT_RSP_ELE_25  =
         ACT_RSP_ELE_26  =
sd_element 2:
         Occurred        = 1
         ErrNum          = 0
         ErrMsg          = ""
         EventFlags      = 1
         EventTimeSec    = 1218641546
         EventTimeUsec   = 371254
         RsrcHndl        = "0x606d 0xffff 0x180031ae
                            0xe300b8db 0x10e9791a 0xc2218e62"
         NodeName        = "c175n06.ppd.pok.ibm.com"
         NumAttrs        = 3
         NumAttrsEvtExpr = 1
         AttrArray       = 11
         ACT_RSP_ELE_11  = "StateSDArray"
         ACT_RSP_ELE_12  = 22
         ACT_RSP_ELE_13  = 20
         ACT_RSP_ELE_14  = "Name"
         ACT_RSP_ELE_15  = 8
         ACT_RSP_ELE_16  = "IBM.Ray01"
         ACT_RSP_ELE_17  = "NodeNameList"
```

```
        ACT_RSP_ELE_18  = 19
        ACT_RSP_ELE_19  = {"c175n06.ppd.pok.ibm.com"}
        ACT_RSP_ELE_20  = 2
        ACT_RSP_ELE_21  = 23
        ACT_RSP_ELE_22  = 25
        ACT_RSP_ELE_23  = 1
        ACT_RSP_ELE_24  = -30396368
        ACT_RSP_ELE_25  = 1
        ACT_RSP_ELE_26  = -30396368
```

Sample output of a **QueryLastEvent** resource action using the name of a condition follows:

```
> runact -s "Name=='/tmp space used'" IBM.Condition QueryLastEvent
Resource Action Response for QueryLastEvent
sd_element 1:
        Occurred        = 1
        ErrNum          = 0
        ErrMsg          = ""
        EventFlags      = 6
        EventTimeSec    = 1218641530
        EventTimeUsec   = 507957
        RsrcHndl        = "0x6009 0xffff 0x180031ae
                           0xe300b8db 0x10bee2df 0x33b20837"
        NodeName        = "c175n06.ppd.pok.ibm.com"
        NumAttrs        = 3
        NumAttrsEvtExpr = 1
        AttrArray       = 11
        ACT_RSP_ELE_11  = "PercentTotUsed"
        ACT_RSP_ELE_12  = 2
        ACT_RSP_ELE_13  = 78
        ACT_RSP_ELE_14  = "Name"
        ACT_RSP_ELE_15  = 8
        ACT_RSP_ELE_16  = "/tmp"
        ACT_RSP_ELE_17  = "NodeNameList"
        ACT_RSP_ELE_18  = 19
        ACT_RSP_ELE_19  = {"c175n06.ppd.pok.ibm.com"}
```

The event information displayed in the preceding **QueryLastEvent** action output contains structured data (SD) elements. Each SD element describes a cached event and is based on the event notification structure described in the *RSCT: RMC Programming Guide*. In the event notification and in the action output, there is a list of attributes and their values that are returned within the event notification. In the action output, the attributes and their values are found in the element fields named **ACT_RSP_ELE**_*nn*. The **NumAttrs** field has the number of attributes in the event. The **NumAttrsEvtExpr** field indicates how many of the attributes are in the event expression, starting with the first attribute. The **AttrArray** field indicates the field number of where the attribute array starts.

Each attribute in the attribute list has at least three fields to describe it. First, there is the name of the attribute (**PercentTotUsed**). Second is its data type (**2=CT_INT32**). The data types are described in the *RSCT: RMC Programming Guide*. The third field contains the value of the attribute (78). Depending on the data type of the attribute, there may be more fields or pointers to other fields. If the attribute data type indicates SD type, instead of an attribute value, the third field is a pointer to the field that begins the description of this structured data. Those fields will contain first the number of elements in the structured data followed by the structured data elements. Similarly, if the attribute data type indicates SD array type, instead of an attribute value, the third field is a pointer to the field that begins the description of this structured data array. Those fields will contain the number of elements in the SD array and the fields that point to each SD.

Use the following file to turn off event caching:

/var/ct/cfg/mc.EReventcache

If the file exists and contains **ENABLE=1**, ERRM will save the last event for each monitored resource. The events will be saved also if the file is not found. If the file exists and contains **ENABLE=0**, the events will not be saved. This text may be preceded by white space but it must not contain any embedded spaces.

To manage the cache, when an event is triggered, ERRM saves the event in its cache based on the condition and the resource for which it is reported. The cache has, at most, one event per condition or resource. If an event is generated and there already is a cached event for this condition and resource, the event replaces the cached event. The cached events are lost if the ERRM is restarted or if the **ResetLastEvent** action is run.

Querying and monitoring CIM properties and associations:

The Common Information Model (CIM) is a data model for organizing computer hardware and software resources into a common object-oriented class hierarchy. Developed and maintained by the Distributed Management Task Force (DMTF), CIM is a conceptual model for extracting management information. In this way, it is similar to the RMC data model.

The CIM resource manager is an RMC resource manager that enables you to use RMC to query or monitor system configuration through CIM classes. The CIM resource manager provides a command (**mkcimreg**) that enables you to register CIM classes with RMC. Once registered, you can:

- query the value of CIM properties using the RMC command **lsrsrc**.
- monitor CIM properties
- query CIM associations using the **lsassocmap** and **lsrsrcassoc** commands.

This documentation describes how to query CIM properties through RMC but does not describe the CIM standard in detail. For complete information on the CIM standard, see DMTF's Web page at http://www.dmtf.org.

Before describing how to query CIM properties through RMC, it is useful to understand the key terminology differences between the CIM and RMC data models, as outlined in Table 54.

Table 54. Terminology differences between the CIM and RMC data models

This CIM term...	Is analogous to this RMC term...	And refers to...
Provider	Resource manager	Processes that can set or return information about a physical or software entity. Defines a number of resource classes (*classes* in CIM terminology).
Class	Resource class	The set of resources (*instances* in CIM terminology) of a common type.
Instance	Resource	The logical abstractions that represent the actual physical or software resources (*managed objects* in CIM terminology).
Property	Attribute	These terms refer to a characteristic of a resource (*instance* in CIM terminology).
Managed Object	Physical or software resource	The actual hardware or software entity that is represented as a resource (*instance* in CIM terminology) by a particular resource manager (*provider* in CIM terminology).
Association	Not applicable. There is no RMC equivalent to a CIM association.	A class that describes the relationship between two other classes.

Related concepts:

"Resource managers provided with RSCT" on page 110
Together with the RMC subsystem, resource managers provide the administrative and monitoring capabilities of RSCT.

Registering CIM classes and providers:

To query a CIM property or association, or monitor a CIM property through RMC, you first need to register the appropriate CIM class and Common Manageability Programming Interface (CMPI) provider with RMC. The CIM resource manager supports only 32-bit CMPI providers.

To register a CIM class and CMPI provider, use the CIM resource manager's **mkcimreg** command. You supply the **mkcimreg** command with a list of Managed Object Format (MOF) files containing either CIM class definitions or provider registration information. The command then outputs files used by the CIM resource manager to enable RMC to work with the CIM classes.

Note that when a CIM class and CMPI method are registered, the CIM methods belonging to that class are mapped to RMC resource actions. For more information, see "Invoking CIM methods" on page 193.

For a current list of CIM classes supported by the CIM resource manager, see the CIM Classes section of the read-only file **/usr/sbin/rsct/cfg/ct_class_ids**. The CIM classes begin with the characters *cimv2*.

Table 55 indicates how to locate the MOF files on Linux nodes and AIX nodes.

Table 55. Locating the CIM MOF files on Linux nodes and AIX nodes

On Linux Nodes	On AIX Nodes
The class and provider MOF files and the provider library files for most of the classes listed are available from the Standards Based Linux Instrumentation for Manageability (SBLIM) Web site at http://sblim.sourceforge.net. The SBLIM providers are also available as part of the SUSE Linux Enterprise Server (SLES) distributions. To access the providers, the **sblim-cmpi** packages must be installed. • On SLES 9, after the **sblim-cmpi** packages are installed, the provider MOF files will be available in the **/usr/share/cmpi/mof** directory. The MOF file names end with `.mof`. • On SLES 10, after the **sblim-cmpi** packages are installed, the provider MOF files will be available in multiple directories named **/usr/share/sblim-cmpi-***provider*, where *provider* stands for the type of provider being registered. The MOF file names end with `.registration`. The provider library directory is **/usr/lib/cmpi**.	The class and provider MOF files and the provider library files for the AIX classes listed in the **ct_class_ids** file are provided with the Pegasus CIM Server and OS base providers package. The Pegasus CIM Server and base providers are part of the AIX Expansion Pack. To use the CIM classes provided by the OS base providers, install the following packages from the expansion pack: **sysmgt.pegasus.cimserver** installs the Pegasus CIM Server filesets in the **/opt/freeware/cimom/pegasus** directory. **sysmgt.pegasus.osbaseproviders** installs the base providers for AIX filesets in the **/usr/pegasus/provider** directory. For more information on the Pegasus CIM Server, see *AIX Common Information Model Guide*, SC23-4942.

In order to query one of the CIM classes listed in **/usr/sbin/rsct/cfg/ct_class_ids**, you will need to register both the CIM class and CIM provider using the **mkcimreg** command. The appropriate class and provider MOF files must also be available on your file system.

Do the following to register CIM classes and providers:

1. Register one or more CIM classes by supplying the **mkcimreg** command with the path(s) to the MOF file(s).
   - On IBM AIX, the MOF files will be located in the directory **/usr/pegasus/provider/mof**.
   - On SUSE Linux Enterprise Server 9 (SLES 9), the MOF files will be located in the directory **/usr/share/cmpi/mof** and will have an extension of `.mof`.
   - On SUSE Linux Enterprise Server 10 (SLES 10), the MOF files will be located in a directory named **/usr/share/sblim-cmpi-***provider* and will have an extension of `.registration`.
   - On other Linux distributions, the MOF files will be located in the directory specified when they were downloaded from the SBLIM Web site.

Note: You cannot register classes that derive from classes that have not yet been registered. When you have a class derived from another, be sure to register the parent class first.

Examples:

a. To register the CIM classes in the MOF file Linux_Base.mof located in the current directory, you would enter:

```
mkcimreg Linux_Base.mof
```

b. To register the CIM classses in the MOF files Linux_Base.mof and Linux_Network.mof, you would enter:

```
mkcimreg Linux_Base.mof Linux_Network.mof
```

You can also use the **-I** flag on the **mkcimreg** command to specify one or more additional directories to be searched for MOF files.

Example: If the MOF files are all located in **/u/jbrady/MOF**, you could enter:

```
mkcimreg -I /u/jbrady/MOF Linux_Base.mof Linux_Network.mof
```

If a class specified on the **mkcimreg** command has already been registered, the **mkcimreg** command will not register the class again and will instead return an error. If you are attempting to register a new version of the class, you can use the **-f** flag to force the class to be registered again.

Example:

```
mkcimreg -f Linux_Base.mof
```

When registering a new version of the class using the **-f** flag, you must also register all subclasses of the upgraded class in order to propagate the changes introduced in the new class to its subclasses. Since the changes propagate from parent class to child class, you must reregister the entire class hierarchy in descending order starting with the topmost parent class and finishing with the lowest-level child class.

2. Register the CIM provider(s) by supplying the **mkcimreg** command with the path to the directory containing the provider library files and the path(s) to the provider MOF file(s).

   You specify the provider library file directory using the **-p** flag. On AIX, the provided library directory is **/usr/pegasus/provider/lib**. On SLES 9 and SLES 10, the provider library directory is **/usr/lib/cmpi**. On other Linux distributions, the provider MOF files will be located in the directory specified when they were downloaded from the SBLIM web site.

   Example: The provider MOF files associated with the Linux_Base.mof and Linux_Network.mof files are Linux_BaseRegistration.mof and Linux_NetworkRegistration.mof. If the library files for these providers were located in **/usr/lib** and the MOF files were in the current directory, you could register them by entering:

   ```
   mkcimreg -p /usr/lib Linux_BaseRegistration.mof Linux_NetworkRegistration.mof
   ```

3. The **mkcimreg** command outputs a number of files which describe new RMC resource classes for the CIM classes defined in the MOF files. In order to detect this new resource class information, you will need to stop the CIM resource manager, and stop and restart the RMC subsystem.

   a. Shut down the CIM resource manager using the **stopsrc** command. Use the **stopsrc** command's **-s** flag to identify the CIM resource manager (IBM.CIMRM).

   ```
   stopsrc -s IBM.CIMRM
   ```

   b. Make sure CIM resource manager has shut down by issuing the **lsrsrc** command. Use the **lsrsrc** command's **-s** flag to indicate that you want the status of the CIM resource manager (IBM.CIMRM).

   ```
   lssrc -s IBM.CIMRM
   ```

   The following output is displayed. Make sure that the output shows the status of the CIM resource manager as `inoperative`. If it is not inoperative, repeat this step.

   ```
   Subsystem         Group            PID     Status
    IBM.CIMRM        rsct_rm          6261    inoperative
   ```

   c. To stop the RMC subsystem, issue the **rmcctrl** command with its **-k** flag. Be aware that this command shuts down RMC. Any RMC-dependent resource monitoring in place at the time is

deactivated. Environments relying on RMC or any of its resource managers for high availability or other critical system functions may become temporarily disabled.

```
rmcctrl -k
```

The **rmcctrl** command is located in **/usr/sbin/rsct/bin**. Add this directory to your PATH, or specify the full path on the command line.

d. Make sure RMC subsystem has shut down by issuing the **lsrsrc** command. Use the **lsrsrc** command's **-s** flag to indicate that you want the status of the RMC subsystem (ctrmc).

```
lssrc -s ctrmc
```

The following output is displayed. Make sure that the output shows the status of the RMC subsystem as `inoperative`. If it is not inoperative, repeat this step.

```
Subsystem         Group            PID     Status
 ctrmc            rsct             6199    inoperative
```

e. To restart the RMC subsystem, issue the **rmcctrl** command with its **-A** flag.

```
rmcctrl -A
```

When you registered the CIM class and its provider, the CIM classes defined in the MOF files were mapped to new RMC resource classes. The RMC resource class name will be a concatenation of the namespace and the CIM class name, for example: **cimv2.Linux_ComputerSystem**. All registered CIM classes are placed in the **root/cimv2** namespace.

Now that you have restarted the RMC subsystem, RMC will have detected these new classes. To verify that the resource classes were created, enter:

```
lsrsrc
```

This command lists all resource classes, so the following output is displayed (the resource classes created for the CIM classes defined in **Linux_Base.mof** and **Linux_Network.mof** are highlighted in this example):

```
class_name
"IBM.Association"
"IBM.AuditLog"
"IBM.AuditLogTemplate"
"IBM.Condition"
"IBM.EthernetDevice"
"IBM.EventResponse"
"IBM.Host"
"IBM.FileSystem"
"IBM.Program"
"IBM.TokenRingDevice"
"IBM.Sensor"
"IBM.PeerDomain"
"IBM.PeerNode"
"IBM.RSCTParameters"
"IBM.NetworkInterface"
"IBM.CommunicationGroup"
"IBM.HostPublic"
"IBM.TieBreaker"
"cimv2.Linux_ComputerSystem"
"cimv2.Linux_OperatingSystem"
"cimv2.Linux_UnixProcess"
"cimv2.Linux_Processor"
"cimv2.Linux_RunningOS"
"cimv2.Linux_OSProcess"
"cimv2.Linux_CSProcessor"
"cimv2.Linux_IPProtocolEndpoint"
"cimv2.Linux_LocalLoopbackPort"
"cimv2.Linux_EthernetPort"
"cimv2.Linux_TokenRingPort"
"cimv2.Linux_CSNetworkPort"
"cimv2.Linux_NetworkPortImplementsIPEndpoint"
```

For detailed syntax information on the **lsrsrc**, **mkcimreg**, and **rmcctrl** commands, see their online man pages, the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Querying and monitoring CIM properties:

Before you can query or monitor a CIM property through RMC, you must have first registered the CIM class and CMPI provider with RMC.

See "Registering CIM classes and providers" on page 185 for more information. Once you have registered the CIM classes and providers and restarted the RMC subsystem, RMC resource classes will be created for the CIM classes. You can query or monitor the properties of any of these new resource classes in the same way you would query or monitor any attribute in RMC.

Querying CIM properties:

To query CIM properties, issue the **lsrsrc** command, supplying it with the resource class name as an argument.

For example, to list the properties for the *cim2.Linux_ComputerSystem* resource class, enter:

```
lsrsrc cimv2.Linux_ComputerSystem
```

The following output is displayed:

```
Resource Persistent Attributes for: cimv2.Linux_ComputerSystem
resource 1:
        NameFormat          = "IP"
        Dedicated           = {0}
        CreationClassName   = "Linux_ComputerSystem"
        Name                = "c175nf01.ppd.pok.ibm.com"
        PrimaryOwnerName    = "root"
        PrimaryOwnerContact = "root@c175nf01.ppd.pok.ibm.com"
        EnabledState        = 2
        OtherEnabledState   = "NULL"
        RequestedState      = 2
        EnabledDefault      = 2
        Status              = "NULL"
        Caption             = "Computer System"
        Description         = "A class derived from ComputerSystem that represents
the single node container of the Linux OS."
        ElementName         = "c175nf01.ppd.pok.ibm.com"
        ActivePeerDomain    = ""
```

For detailed attribute definition information, use the **lsrsrcdef** command.

Example: The following command returns detailed attribute information for the *cimv2.Linux_ComputerSystem* resource class:

```
lsrsrcdef -e cimv2.Linux_ComputerSystem
```

The following output is displayed:

```
Resource Persistent Attribute Definitions for: cimv2.Linux_ComputerSystem
attribute 1:
        program_name  = "NameFormat"
        display_name  = "NameFormat"
        group_name    = "description is not available"
        properties    = {"option_for_define","selectable","public"}
        description   = "The ComputerSystem object and its derivatives are Top L
evel Objects of CIM. They provide the scope for numerous components. Having uniq
ue System keys is required. The NameFormat property identifies how the ComputerS
ystem Name is generated. The NameFormat ValueMap qualifier defines the various m
echanisms for assigning the name. Note that another name can be assigned and use
d for the ComputerSystem that better suit a business, using the inherited Elemen
```

```
tName property."
        attribute_id  = 0
        group_id      = 0
        data_type     = "char_ptr"
        variety_list  = {[1,1]}
        variety_count = 1
        default_value = ""
attribute 2:
        program_name  = "OtherIdentifyingInfo"
        display_name  = "OtherIdentifyingInfo"
        group_name    = "description is not available"
        properties    = {"option_for_define","selectable","public"}
        description   = "OtherIdentifyingInfo captures additional data, beyond S
ystem Name information, that could be used to identify a ComputerSystem. One exa
mple would be to hold the Fibre Channel World-Wide Name (WWN) of a node. Note th
at if only the Fibre Channel name is available and is unique (able to be used as
 the System key), then this property would be NULL and the WWN would become the
System key, its data placed in the Name property."
        attribute_id  = 1
        group_id      = 0
        data_type     = "char_ptr_array"
        variety_list  = {[1,1]}
        variety_count = 1
        default_value = {""}
attribute 3:
        program_name  = "IdentifyingDescriptions"
⋮
```

For detailed syntax information on the **lsrsrc** and **lsrsrcdef** commands, see their online man pages. For detailed syntax information, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Monitoring CIM properties:

When you register a CIM class and provider with RMC, RMC resource classes are created for the CIM classes.

Instances of a CIM class are represented as resources of the resource class in RMC, and CIM properties are represented by persistent attributes of the resource. Typically, persistent attributes in RMC represent values that are generally unchanging and so are not usually monitored. However, in the case of CIM properties represented as persistent attributes, the values may or may not be changing, depending on the characteristics of the underlying provider.

You monitor a CIM property (attribute) the same way you would monitor any attribute in RMC — by creating a condition (as described in “Creating a condition” on page 157) that identifies the attribute value to monitor. An event expression in the condition specifies what changes in the attribute value will indicate an event of interest. Once defined, the condition can then be associated with a response (described in “Creating a condition/response association” on page 140) that describes how RMC should respond when the condition event occurs. Finally, you can start monitoring the condition using the **mkcondresp** command (as described in “Starting condition monitoring” on page 142.

For example, the following **mkcondition** command creates a condition to monitor the PercentageSpaceUsed property of the CIM class Linux_NFS. The RMC resource class name, as with all registered CIM classes, is a concatenation of the namespace *cimv2* and the CIM class name.

```
mkcondition PercentageSpaceUsed -r cimv2.Linux_NFS -e "PercentageSpaceUse > 66"
     -E "PercentageSpaceUse < 65" -S w "File System Full"
```

The preceding command creates a condition named `"File System Full"`. To associate this new condition with the predefined response `"E-mail root any time"`, you would issue the following command:

```
mkcondresp "File System Full" "E-mail root any time"
```

To start monitoring the condition, you would issue the following command:

```
startcondresp "File System Full"
```

For detailed syntax information on the **mkcondition**, **mkcondresp**, and **startcondresp** commands, see their online man pages.

Querying CIM associations:

Before you can query CIM associations through RMC, you must have first registered the CIM association class, the CIM classes it associates, and the provider with RMC.

See "Registering CIM classes and providers" on page 185 for more information. Once you have registered the CIM classes and providers and restarted the RMC subsystem, RMC resources classes will be created for CIM classes. An RMC resource class will be created for an association class if the association's CIM class provider is also an instance provider. An association class describes the relationship between two other classes. To query CIM associations, you can use:

- the **lsassocmap** command to display the associations available
- the **lsrsrcassoc** command to list resources that are associated with a particular class

Typically, you will use the **lsassocmap** to display an association map (an overview of the associations available and the classes that each associates). Once you know which classes have associated classes, you can issue the **lsrsrcassoc** command to display information about resources of an associated class.

If a CIM association class' provider is not also an instance provider, no RMC resource class will have been created for the CIM association class. However, information for the CIM class will still be displayed by the **lsassocmap** and **lsrsrcassoc** commands.

To display the association map for all association classes, enter the **lsassocmap** command with no command flags.

```
lsassocmap
```

The following output is displayed:

```
Association Class         Role 1         Associator 1          Role 2        Associator 2         Node
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor      davros.pok.ibm.com
cimv2.Linux_CSNetworkPort GroupComponent Linux_ComputerSystem  PartComponent Linux_NetworkPort    davros.pok.ibm.com
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor      davros.pok.ibm.com
cimv2.Linux_OSProcess     GroupComponent Linux_OperatingSystem PartComponent Linux_UnixProcess    davros.pok.ibm.com
cimv2.Linux_RunningOS     Antecedent     Linux_OperatingSystem Dependent     Linux_ComputerSystem davros.pok.ibm.com

cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor      omega.pok.ibm.com
cimv2.Linux_CSNetworkPort GroupComponent Linux_ComputerSystem  PartComponent Linux_NetworkPort    omega.pok.ibm.com
```

The columns in the output indicate the following:

Association Class
The CIM Association class that describes the relationship between two classes.

Role 1 The role that the class identified in the **Associator 1** column on this output plays in the association.

Associator 1
The name of one class (one endpoint) in the association.

Role 2 The role that the class identified in the **Associator 2** column on this output plays in the association.

Associator 2
The name of another class (the other endpoint) in the association class or provider.

Node The node containing the association class and provider.

To display the associations for a particular association class, use the **lsassocmap** command's **-c** flag. The following command lists only the associations for the *Lunux_CSProcessor* association class.

```
lsassocmap -c Linux_CSProcessor
```

The following output is displayed:

```
Association Class         Role 1         Associator 1          Role 2        Associator 2      Node
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   davros.pok.ibm.com
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   davros.pok.ibm.com

cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   omega.pok.ibm.com
```

To display only the associations that include a particular resource class as one of the endpoints, specify the class name on the **lsassocmap** command line. The following command lists only the associations that include the resource class *Linux_ComputerSystem* as one of its associators.

```
lsassocmap Linux_ComputerSystem
```

The following output is displayed:

```
Association Class         Role 1         Associator 1          Role 2        Associator 2      Node
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   davros.pok.ibm.com
cimv2.Linux_CSNetworkPort GroupComponent Linux_ComputerSystem  PartComponent Linux_NetworkPort davros.pok.ibm.com
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   davros.pok.ibm.com

cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   omega.pok.ibm.com
cimv2.Linux_CSNetworkPort GroupComponent Linux_ComputerSystem  PartComponent Linux_NetworkPort omega.pok.ibm.com
```

Any number of classes can be listed on the **lsassocmap** command line. Only associations that include any of the specified classes will be listed. For example:

```
lsassocmap Linux_ComputerSystem Linux_UnixProcess
```

```
Association Class         Role 1         Associator 1          Role 2        Associator 2      Node
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   davros.pok.ibm.com
cimv2.Linux_CSNetworkPort GroupComponent Linux_ComputerSystem  PartComponent Linux_NetworkPort davros.pok.ibm.com
cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   davros.pok.ibm.com
cimv2.Linux_OSProcess     GroupComponent Linux_OperatingSystem PartComponent Linux_UnixProcess davros.pok.ibm.com

cimv2.Linux_CSProcessor   GroupComponent Linux_ComputerSystem  PartComponent Linux_Processor   omega.pok.ibm.com
cimv2.Linux_CSNetworkPort GroupComponent Linux_ComputerSystem  PartComponent Linux_NetworkPort omega.pok.ibm.com
```

Once you know which classes have associations, you can use the **lsrsrcassoc** command to list resource information about one class' associated class or classes. To list information for all resources of classes associated with any resource of the Linux_OperatingSystem class, you would enter:

```
lsrsrcassoc Linux_OperatingSystem
```

You can specify either class in an association when issuing the **lsrsrcassoc** command. The **lsrsrcassoc** command will then display information for resources of the specified class' associated class or classes. For clarity, we will refer to the class specified on the **lsrsrcassoc** command as the *source class* and any class associated with it as a *destination class*.

When you supply the **lsrsrcassoc** command with only a source class name, it will list all property values for all resources of any destination class. You can use the **lsrsrcassoc** command's additional parameters and flags to display only a subset of the information, as described in Table 56 on page 192.

Table 56. Using the **lsrsrcassoc** *command to list resource information*

To list the following information...	Specify the following on the lsrsrcassoc command...	Examples
Only those resources of the destination class(es) that are linked to the source class through a particular Association Class	The Association Class using the **-c** flag	To list only those resources tied to the source class Linux_OperatingSystem through the Association Class Linux_OSProcess, you would enter: **lsrsrcassoc -c Linux_OSProcess Linux_OperatingSystem**
All resources of the destination class(es) that are associated with only those resources of the source class that match a particular selection string	The selection string used to filter resources of the source class using the **-s** flag	To list all resources associated with only those resources of the source class Linux_OperatingSystem that match the selection string 'Name=~"davros"', you would enter: **lsrsrcassoc -s 'Name=~"davros"' Linux_OperatingSystem**
Only those resources of the destination class(es) that match a particular selection string	The selection string used to filter resources of the destination class using the **-S** flag	To list only those resources of the destination class (associated with the source class Linux_OperatingSystem) that match the selection string 'Name=~"emacs"', you would enter: **lsrsrcassoc -S 'Name=~"emacs" Linux_OperatingSystem**
Only a subset of CIM property values for the resources of the destination class	One or more CIM properties following the source class name on the **lsrsrcassoc** command	To specify that only the values of the CIM properties *handle* and *parameters* should appear in the output, you would enter: **lsrsrcassoc Linux_OperatingSystem Handle Parameters**

The following command uses the flags and parameters described in Table 56 to limit the output of the **lsrsrcassoc** command to show only the information of interest. Specifically, it shows only selected properties of resources of the destination class linked with the source class Linux_OperatingSystem through the Association Class Linux_OSProcess. Selection strings are used to filter resources of both the source class and destination class so that only Emacs processes for the node named *davros.pok.ibm.com* are listed.

```
lsrsrcassoc -a Linux_OSProcess -s 'Name=~"davros"' -S 'Name=~"emacs" Linux_OperatingSystem Handle Parameters
```

The output from the command is:

```
Resource Persistent Attributes for cimv2.Linux_UnixProcess
resource 1:
Handle = "2781"
Parameters = {"emacs", "-u", "foo.C"}
resource 2:
Handle = "2782"
Parameters = {"emacs", "bar.C"}
resource 3:
Handle = "2783"
Parameters = {"emacs","foo_bar.C"}
resource 4:
Handle = "2784"
Parameters = {"emacs","bar_foo.C"}
resource 5:
Handle = "2785"
Parameters = {"emacs","CIMRC.C"}
resource 6:
Handle = "26994"
Parameters = {"emacs","lsassocmap.pl"}
```

Targeting nodes:

The **lsassocmap** and **lsrsrcassoc** commands are affected by the environment variables CT_CONTACT and CT_MANAGEMENT_SCOPE. The CT_CONTACT environment variable indicates a node whose RMC daemon will carry out the command request (by default, the local node on which the command is issued). The CT_MANAGEMENT_SCOPE indicates the management scope — either local scope, peer domain scope, or management domain scope.

Invoking CIM methods:

When a CIM class and CMPI method provider have been registered with RMC, the CIM methods belonging to that class are mapped to RMC resource actions.

As a result, you can use the RMC **lsactdef** command to list the actions that are available for a given CIM class, as well as the RMC **runact** command to run those actions.

CIM class and CMPI method provider registration with RMC is described in "Registering CIM classes and providers" on page 185.

Note: For Linux, methods are supported by only a subset of the SUSE Linux Enterprise Server 9 providers. For AIX, there is no method support among the base providers.

Catching SNMP traps on Linux nodes:

The ability to catch SNMP trap messages is only available on Linux nodes. This capability is not available as part of the AIX implementation of RSCT.

The Simple Network Management Protocol (SNMP), a standard operations and maintenance protocol, uses trap-directed notification for receiving information about managed devices. Instead of polling each managed device, which can be resource intensive, an agent on a managed device can send unsolicited messages when events of interest occur. These unsolicited messages are known as SNMP traps.

If you have an SNMP-managed network, you can use RMC on Linux nodes to catch SNMP traps. You can use RMC's event management capabilities to respond to the trap message as you would respond to a monitored event in RMC. The SNMP trap information is also entered into the audit log.

Do the following to catch SNMP traps:

1. Run the **cfgrmcsnmp** command. This command will configure the node to receive SNMP traps.

   ```
   cfgrmcsnmp
   ```

   The **cfgrmcsnmp** command is located in **/usr/sbin/rsct/install/bin**. Add this directory to your PATH, or specify the full path on the command line.

   When a node is configured to receive SNMP traps, a sensor object named SNMPTrap is added to the RMC subsystem. When an SNMP trap is received, the String dynamic attribute of the SNMPTrap sensor object will be updated to reflect the trap information. The String dynamic attribute will contain the trap origin, type, and value information separated by newline characters. For example, issuing the following command to generate a trap:

   ```
   snmptrap -v 2c -c public localhost '' 0 0 s "Hello, this is an SNMP trap."
   ```

   would cause the String attribute of the SNMPTrap sensor to be updated. Using the generic RMC command **lsrsrc**, you can display the trap information. The command:

   ```
   lsrsrc -s "Name='SNMPTrap'" IBM.Sensor String
   ```

   Would return:

   ```
   Resource Persistent Attributes for IBM.Sensor
   resource 1:
           String = SNMP Trap from localhost.localdomain (127.0.0.1)\nTrap Ty
   pe: zeroDotZero\nOID: zeroDotZero VALUE: Hello, this is an SNMP trap.
   ```

2. A predefined condition named SNMP trap detected will have been created when RSCT was installed. Use the **mkcondresp** command to associate this condition with a response of your choice. You can use one of the predefined responses, or you can create one of your own as described in "Creating a response" on page 168.

The following example associates the SNMP trap detected condition with the predefined response Broadcast details of event any time.

```
mkcondresp "SNMP trap detected" "Broadcast details of event any time"
```

3. Start condition monitoring (SNMP trap detection) using the **startcondresp** command:

```
startcondresp "SNMP trap detected"
```

To verify that the condition is being monitored, you can use the **lscondition** command:

```
lscondition
```

Output is similar to:

```
Displaying condition information:
Name                 MonitorStatus
"SNMP trap detected" "Monitored"
```

To later stop SNMP trap detection, you can use the **stopcondresp** command:

```
stopcondresp "SNMP trap detected"
```

To verify that the condition is no longer being monitored, you can use the **lscondition** command:

```
lscondition
```

Output is similar to:

```
Displaying condition information:
Name                 MonitorStatus
"SNMP trap detected" "Not monitored"
```

To unconfigure the ability to detect SNMP traps on the node, enter the **cfgrmcsnmp** command with its **-u** flag:

```
cfgrmcsnmp -u
```

For detailed syntax information on the **cfgrmcsnmp**, **mkcondresp**, **startcondresp**, and **stopcondresp** commands, see their online man pages. For detailed syntax information, see the *Technical Reference: RSCT for AIX* or *Technical Reference: RSCT for Multiplatforms* guides.

Locking and unlocking conditions, compound conditions, responses, and condition/response links:

Conditions, compound conditions, responses, and condition/response links can be locked to prevent user modification or removal. A locked condition, compound condition, response, or condition/response link cannot be modified or removed until it is unlocked.

For this reason, the following commands for manipulating conditions, compound conditions, responses, and condition/response links may fail to make the expected change if the resource you are trying to manipulate with the command is locked. Instead, an error will be generated informing you that the condition, compound conditions, response, or condition/response link is locked. The commands that will fail to act upon a particular locked resource are:

- the **chccondition** command which modifies a compound condition. A locked compound condition cannot be modified.
- the **chcondition** command which modifies a condition. A locked condition cannot be modified.
- the **chresponse** command which modifies a response. A locked response cannot be modified.
- the **rmccondition** command which removes a compound condition. A locked compound condition cannot be removed.
- the **rmcondition** command which removes a condition. A locked condition cannot be removed.
- the **rmcondresp** command which deletes the link between a condition and response. A locked condition/response link cannot be removed.

- the **rmresponse** command which removes a response. A locked response cannot be removed.
- the **startcondresp** command which starts monitoring a condition that has one or more linked responses. If the condition/response link is locked, you will not be able to start monitoring.
- the **stopcondresp** command which stops monitoring a condition that has one or more linked responses. If the condition/response link is locked, you will not be able to stop monitoring.

System software that uses RSCT may lock certain monitoring resources that are considered vital for the system software to work properly. Similarly, as a system administrator, you may choose to lock certain monitoring resources that you consider vital in order to prevent accidental modification or removal.

Two flags (**-L** and **-U**) are provided on a number of Event Response Resource Manager commands to enable you to lock and unlock monitoring resources. The **-L** flag locks the condition, compound condition, response, or condition/response link, while the **-U** flag unlocks it.

Before using the **-U** flag, be aware that if a particular condition, compound condition, response, or condition/response link has been locked, this may be because it is essential for system software to work properly. For this reason, exercise caution before unlocking a condition, compound condition, response, or condition/response link. In general, if you do not know why the monitoring resource is locked, do not unlock it.

Locking or unlocking a condition:

To lock or unlock a condition, use the **-L** and **-U** flags on the **chcondition** command. When using either of these flags, no other operation can be performed by the **chcondition** command.

The syntax is:

chcondition {**-L** | **-U**} *condition*[**:***node_name*]

Note: If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server.

Examples:

- If you have created a condition named `/usr space used` and now want to lock it to prevent accidental modification or removal, you would enter:

  ```
  chcondition -L "/usr space used"
  ```

- To unlock this condition, you would enter:

  ```
  chcondition -U "/usr space used"
  ```

Locking or unlocking a response:

To lock or unlock a response, use the **-L** and **-U** flags on the **chresponse** command. When using either of these flags, no other operation can be performed by the **chresponse** command.

The syntax is:

chresponse {**-L** | **-U**} *response*[**:***node_name*]

Note: If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server.

Examples:

- If you have created a response named `log info to /tmp/EventLog` and now want to lock it to prevent accidental modification or removal, you would enter:

  ```
  chresponse -L "log info to /tmp/EventLog"
  ```

- To unlock this response, you would enter:

  ```
  chresponse -U "log info to /tmp/EventLog"
  ```

Locking or unlocking a condition/response link:

To lock or unlock a condition/response link, use the **-L** and **-U** flags on either the **rmcondresp** command, the **startcondresp** command, or the **stopcondresp** command.

The **-L** and **-U** flags perform the exact same operation regardless of which of the commands you use. No other operation can be performed by these commands when you use the **-L** or **-U** flag.

The syntax for locking or unlocking a condition/response link using the **rmcondresp** command is:

rmcondresp {**-L** | **-U**} *condition*[**:***node_name*] *response*

The syntax for locking or unlocking a condition/response link using the **startcondresp** command is:

startcondresp {**-L** | **-U**} *condition*[**:***node_name*] *response*

The syntax for locking or unlocking a condition/response link using the **stopcondresp** command is:

stopcondresp {**-L** | **-U**} *condition*[**:***node_name*] *response*

Note: If you are running the command on the management server, do not specify the management server as the targeted node using the **:node_name** syntax. Only managed nodes can be targeted this way. The management server resources are used automatically when the commands are run on the management server.

Examples:

- Suppose that you have created a link between a condition `/usr space used` and a response `log info to /tmp/EventLog` and started monitoring. To prevent a user from accidentally stopping monitoring, you could lock this condition/response link. Since the **-L** flag is provided on the **rmcondresp** command, the **startcondresp** command, and the **stopcondresp** command, any of the following commands will lock the condition/response link.

  ```
  rmcondresp -L "/usr space used" "log info to /tmp/EventLog"
  startcondresp -L "/usr space used" "log info to /tmp/EventLog"
  stopcondresp -L "/usr space used" "log info to /tmp/EventLog"
  ```

- Similarly, any of the following commands will unlock the condition/response link so it can be stopped, started, or removed.

  ```
  rmcondresp -U "/usr space used" "log info to /tmp/EventLog"
  startcondresp -U "/usr space used" "log info to /tmp/EventLog"
  stopcondresp -U "/usr space used" "log info to /tmp/EventLog"
  ```

Using expressions to specify condition events and command selection strings

An expression in RMC is similar to a C language statement or the WHERE clause of an SQL query. It is composed of variables, operators, and constants.

The C and SQL syntax styles may be intermixed within a single expression.

There are two types of expressions you can specify on certain RMC and ERRM commands. One type is an *event expression*, such as the event expressions and rearm event expressions that you define for conditions using the **mkcondition** or **chcondition** command. Event expressions are described in "Event expressions" on page 120 and "Rearm event expressions" on page 121.

The other type of expression you can specify on certain RMC and ERRM commands is a *selection string expression*. A number of commands enable you to specify a selection string expression that restricts the command action in some way. Table 57 summarizes the commands that accept a selection string expression. For general information about how the selection strings are used by these commands, see the topics referenced in the table.

Table 57. Commands whose actions you can restrict using selection strings

Command	Purpose	Selection string expression action	For more information, see...
chcondition	Changes the attributes of a condition. The condition monitors an attribute of one or more resources of a specified class.	Restricts the command to a subset of the resources in the resource class. The selection string expression filters the available resources by one or more persistent attributes of the resource class. The defined condition will monitor the attribute for only those resources that match the selection string.	"Modifying a condition" on page 164
chrsrc	Changes persistent attribute values of a resource within a specified resource class.	Identifies the resource within the resource class. The selection string expression filters the available resources by one or more persistent attributes of the resource class.	*Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides
lsaudrec	Lists records from the audit log.	Filters the audit log so that only records that match the selection string are listed. The selection string expression filters the audit log using one or more record field names.	"Using the audit log to track monitoring activity" on page 145
lsrsrc	Lists resources of a resource class.	Restricts the command to a subset of the resources in the resource class. The selection string expression filters the available resources by one or more persistent attributes of the resource class. Only the resource(s) that match the selection string will be listed.	*Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides
mkcondition	Creates a new condition. The condition monitors an attribute of one or more resources of a specified class.	Restricts the command to a subset of the resources in the resource class. The selection string expression filters the available resources by one or more persistent attributes of the resource class. The defined condition will monitor the attribute for only those resources that match the selection string.	"Creating a condition" on page 157
rmaudrec	Removes records from the audit log.	Specifies the set of records in the audit log that should be removed. The selection string identifies the records using one or more record field names. Only records that match the selection string are removed.	"Deleting entries from the audit log" on page 150
rmrsrc	Removes resources of a specified resource class.	Restricts the command to a subset of the resources in the resource class. The selection string expression filters the available resources by one or more persistent attributes of the resource class. Only the resource(s) that match the selection string will be removed.	*Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides

SQL restrictions

SQL syntax is supported for selection strings.

Table 58 relates the RMC terminology to SQL terminology.

Table 58. Relationship of RMC terminology to SQL terminology

RMC terminology	SQL terminology
attribute name	column name
selection string	WHERE clause
operators	predicates, logical connectives
resource class	table

Although SQL syntax is generally supported in selection strings, the following restrictions apply.

- Only a single table may be referenced in an expression.
- Queries may not be nested.
- The IS NULL predicate is not supported because there is no concept of a NULL value.
- The period (.) operator is not a table separator (for example, table.column). Rather, in this context, the period (.) operator is used to separate a field name from its containing structure name.
- The pound sign (#) is hard-coded as the escape character within SQL pattern strings.
- All column names are case sensitive.
- All literal strings must be enclosed in either single or double quotation marks. Bare literal strings are not supported because they cannot be distinguished from column and attribute names.

Supported base data types

The term *variable* is used in this context to mean the column name or attribute name in an expression.

Table 59 lists the base data types supported by the RMC subsystem for variables and constants in an expression.

Table 59. Supported base data types for variables and constants in an expression

Symbolic name	Description
CT_INT32	Signed 32–bit integer
CT_UINT32	Unsigned 32–bit integer
CT_INT64	Signed 64–bit integer
CT_UINT64	Unsigned 64–bit integer
CT_FLOAT32	32–bit floating point
CT_FLOAT64	64–bit floating point
CT_CHAR_PTR	Null-terminated string
CT_BINARY_PTR	Binary data – arbitrary-length block of data
CT_RSRC_HANDLE_PTR	Resource handle – an identifier for a resource that is unique over space and time (20 bytes)

Aggregate data types

In addition to the base data types, aggregates of the base data types may be used as well.

The first aggregate data type is similar to a structure in C in that it can contain multiple fields of different data types. This aggregate data type is referred to as *structured data* (SD). The individual fields in the structured data are referred to as *structured data elements* or *elements*. Each element of a structured data type may have a different data type which can be one of the base types described in "Supported base data types" or any of the array types discussed in the next paragraph, except for the structured data array.

The second aggregate data type is an array. An array contains zero or more values of the same data type, such as an array of CT_INT32 values. Each of the array types has an associated enumeration value

(CT_INT32_ARRAY, CT_UINT32_ARRAY). Structured data may also be defined as an array but is restricted to have the same elements in every entry of the array.

Data types that can be used for literal values

Each of these base data types can specify literal values.

Literal values can be specified for each of the base data types as follows:

Array An array or list of values may be specified by enclosing variables or literal values, or both, within braces {} or parentheses () and separating each element of the list with a comma. For example: { 1, 2, 3, 4, 5 } or ("abc", "def", "ghi").

Entries of an array can be accessed by specifying a subscript as in the C programming language. The index corresponding to the first element of the array is always zero; for example, List [2] references the third element of the array named List. Only one subscript is allowed. It may be a variable, a constant, or an expression that produces an integer result. For example, if List is an integer array, then List[2]+4 produces the sum of 4 and the current value of the third entry of the array.

Binary Data

A binary constant is defined by a sequence of hexadecimal values, separated by white space. All hexadecimal values comprising the binary data constant are enclosed in double quotation marks. Each hexadecimal value includes an even number of hexadecimal digits, and each pair of hexadecimal digits represents a byte within the binary value. For example:

```
"0xabcd 0x0102030405060708090a0b0c0d0e0f1011121314"
```

Character Strings

A string is specified by a sequence of characters surrounded by single or double quotation marks (you can have any number of characters, including none). Any character may be used within the string except the null '\0' character. Double quotation marks and backslashes may be included in strings by preceding them with the backslash character.

Floating Types

These types can be specified by the following syntax:

- A leading plus (+) or minus (-) sign
- One or more decimal digits
- A radix character, which at this time is the period (.) character
- An optional exponent specified by the following:
  - A plus (+) or minus (-) sign
  - The letter 'E' or 'e'
  - A sequence of decimal digits (0–9)

Integer Types

These types can be specified in decimal, octal, or hexadecimal format. Any value that begins with the digits 1-9 and is followed by zero or more decimal digits (0-9) is interpreted as a decimal value. A decimal value is negated by preceding it with the character '-'. Octal constants are specified by the digit 0 followed by 1 or more digits in the range 0-7. Hexadecimal constants are specified by a leading 0 followed by the letter x (uppercase or lowercase) and then followed by a sequence of one or more digits in the range 0–9 or characters in the range a–f (uppercase or lowercase).

Resource Handle

A fixed-size entity that consists of two 16-bit and four 32-bit words of data. A literal resource handle is specified by a group of six hexadecimal integers. The first two values represent 16-bit integers and the remaining four each represent a 32-bit word. Each of the six integers is separated by white space. The group is surrounded by double quotation marks. The following is an example of a resource handle:

```
"0x4018 0x0001 0x00000000 0x0069684c 0x00519686 0xaf7060fc"
```

Structured Data

Structured data values can be referenced only through variables. Nevertheless, the RMC command line interface displays structured data (SD) values and accepts them as input when a resource is defined or changed. A literal SD is a sequence of literal values, as defined in "Data types that can be used for literal values" on page 199, that are separated by commas and enclosed in square brackets. For example, ['abc',1,{3,4,5}] specifies an SD that consists of three elements: (a) the string 'abc', (b) the integer value 1, and (c) the three-element array {3,4,5}.

Variable names refer to values that are not part of the expression but are accessed while evaluating the expression. For example, when RMC processes an expression, the variable names are replaced by the corresponding persistent or dynamic attributes of each resource.

Entries of an array may be accessed by specifying a subscript as in 'C'. The index corresponding to the first element of the array is always 0 (for example, List[2] refers to the third element of the array named List). Only one subscript is allowed. It may be a variable, a constant, or an expression that produces an integer result. A subscripted value may be used wherever the base data type of the array is used. For example, if List is an integer array, then "List[2]+4" produces the sum of 4 and the current value of the third entry of the array.

The elements of a structured data value can be accessed by using the following syntax:

```
<variable name>.<element name>
```

For example, `a.b`

The variable name is the name of the table column or resource attribute, and the element name is the name of the element within the structured data value. Either or both names may be followed by a subscript if the name is an array. For example, `a[10].b` refers to the element named b of the 11th entry of the structured data array called a. Similarly, `a[10].b[3]` refers to the fourth element of the array that is an element called b within the same structured data array entry `a[10]`.

How variable names are handled

Variable names refer to values that are not part of an expression but are accessed while evaluating the expression.

When used to select a resource, the variable name is a persistent attribute. When used to generate an event, the variable name is usually a dynamic attribute, but may be a persistent attribute. When used to select audit records, the variable name is the name of a field within the audit record.

A variable name is restricted to include only 7-bit ASCII characters that are alphanumeric (a-z, A-Z, 0-9) or the underscore character (_). The name must begin with an alphabetic character.

When the expression is used by the RMC subsystem for an event or a rearm event, the name can have a suffix that is the '@' character followed by 'P', which refers to RMC's previous observation of the attribute value. Because RMC observes attribute values periodically and keeps track of the previously observed value, you can use this syntax to compare the currently observed value with the previously observed value. For example, the following event expression would trigger an event if the average number of processes on the run queue has increased by 50% or more between observations:

```
(ProcRunQueue - ProcRunQueue@P) >= (ProcRunQueue@P * 0.5)
```

Operators that can be used in expressions

Constants and variables may be combined by an operator to produce a result that, in turn, may be used with another operator. The resulting data type of the expression must be a scalar integer or floating-point value. If the result is 0, the expression is considered to be FALSE; otherwise, it is TRUE.

Note: Blanks are optional around operators and operands unless their omission causes an ambiguity. An ambiguity typically occurs only with the word form of an operator (such as AND, OR, IN, and LIKE).

With these operators, a blank or separator, such as a parenthesis or bracket, is required to distinguish the word operator from an operand. For example, `aANDb` is ambiguous. It is unclear if this is intended to be the variable name aANDb or the variable names a and b combined with the AND operator. It is actually interpreted by the application as a single variable name aANDb. With non-word operators (such as +, -, =, &&, and so on) this ambiguity does not exist, and therefore blanks are optional.

Table 60 summarizes the set of operators that can be used in expressions.

Table 60. Operators that can be used in expressions

Operator	Description	Left data types	Right data types	Example	Notes
+	Addition	float, integer	float, integer	"1+2" results in 3	
-	Subtraction	float, integer	float, integer	"1.0-2.0" results in -1.0	
*	Multiplication	float, integer	float, integer	"2*3" results in 6	
/	Division	float, integer	float, integer	"2/3" results in 1	
-	Unary minus	None	float, integer	"-abc"	
+	Unary plus	None	float, integer	"+abc"	
..	Range	integer	integer	"1..3" results in 1,2,3	This is a shorthand notation for all integers between and including the two values.
%	Modulo	integer	integer	"10%2" results in 0	
|	Bitwise OR	integer	integer	"2|4" results in 6	
&	Bitwise AND	integer	integer	"3&2" results in 2	
~	Bitwise complement	None	integer	~0x0000ffff results in 0xffff0000	
^	Exclusive OR	integer	integer	0x0000aaaa^0x0000ffff results in 0x00005555	
>>	Right shift	integer	integer	0x0fff>>4 results in 0x00ff	
<<	Left shift	integer	integer	"0x0ffff<<4" results in 0xffff0	
== =	Equality	All (except float and SD)	All (except float and SD)	"2==2" results in 1 "2=2" results in 1	The result is 0 (FALSE) or 1 (TRUE).
!= <>	Inequality	All (except SD)	All (except SD)	"2!=2" results in 0 "2<>2" results in 0	The result is 0 (FALSE) or 1 (TRUE).
>	Greater than	float, integer	float, integer	"2>3" results in 0	The result is 0 (FALSE) or 1 (TRUE).
>=	Greater than or equal	float, integer	float, integer	"4>=3" results in 1	The result is 0 (FALSE) or 1 (TRUE).
<	Less than	float, integer	float, integer	"4<3" results in 0	The result is 0 (FALSE) or 1 (TRUE).
<=	Less than or equal	float, integer	float, integer	"2<=3" results in 1	The result is 0 (FALSE) or 1 (TRUE).
=~	Pattern match	string	string	"abc"=~"a.*" results in 1	The right operand is interpreted as an extended regular expression. To use this operator in an expression, the locales of the nodes running the RMC daemon must be using either Unicode Transfer Format-8 (UTF-8) encoding, a codeset that matches UTF-8, or C locale encoding. If multiple nodes are involved, the encoding must be consistent across all nodes.

Table 60. Operators that can be used in expressions (continued)

Operator	Description	Left data types	Right data types	Example	Notes
!~	Not pattern match	string	string	"abc"!~"a.*" results in 0	The right operand is interpreted as an extended regular expression. To use this operator in an expression, the locales of the nodes running the RMC daemon must be using either Unicode Transfer Format-8 (UTF-8) encoding, a codeset that matches UTF-8, or C locale encoding. If multiple nodes are involved, the encoding must be consistent across all nodes.
=? LIKE like	SQL pattern match	string	string	"abc"=? "a%" results in 1	The right operand is interpreted as an SQL pattern.
!? NOT LIKE not like	Not SQL pattern match	string	string	"abc"!? "a%" results in 0	The right operand is interpreted as an SQL pattern.
|< IN in	Contains any	All (except SD)	All (except SD)	"{1..5}|<{2,10}" results in 1	The result is 1 (TRUE) if the left operand contains any value from the right operand.
>< NOT IN not in	Contains none	All (except SD)	All (except SD)	"{1..5}><{2,10}" results in 0	The result is 1 (TRUE) if the left operand contains no value from the right operand.
&<	Contains all	All (except SD)	All (except SD)	"{1..5}&<{2,10}" results in 0	The result is 1 (TRUE) if the left operand contains all values from the right operand.
|| OR or	Logical OR	integer	integer	"(1<2)||(2>4)" results in 1	The result is 0 (FALSE) or 1 (TRUE).
&& AND and	Logical AND	integer	integer	"(1<2)&&(2>4)" results in 0	The result is 0 (FALSE) or 1 (TRUE).
! NOT not	Logical NOT	None	integer	"!(2==4)" results in 1	The result is 0 (FALSE) or 1 (TRUE).

When integers of different signs or size are operands of an operator, standard C-style casting is implicitly performed. When an expression with multiple operators is evaluated, the operations are performed in the order defined by the precedence of the operator. The default precedence can be overridden by enclosing the portion or portions of the expression to be evaluated first in parentheses (). For example, in the

expression "1+2*3", multiplication is normally performed before addition to produce a result of 7. To evaluate the addition operator first, use parentheses as follows: "(1+2)*3". This produces a result of 9.

Table 61 shows the default precedence order for operators in expressions. All operators in the same table cell have the same or equal precedence.

Table 61. Operator precedence in expressions, from highest to lowest

Operators	Description
.	Structured data element separator
~	Bitwise complement
! NOT not	Logical not
-	Unary minus
+	Unary plus
*	Multiplication
/	Division
%	Modulo
+	Addition
-	Subtraction
<<	Left shift
>>	Right shift
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal

Table 61. Operator precedence in expressions, from highest to lowest (continued)

Operators	Description
==	Equality
!=	Inequality
=? LIKE like	SQL match
!?	SQL not match
=~	Regular expression match
!~	Regular expression not match
?=	Regular expression match (compact)
|< IN in	Contains any
>> NOT IN not in	Contains none
&<	Contains all
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise inclusive OR
&&	Logical AND
||	Logical OR
,	List separator

Pattern matching

Two types of pattern matching are supported; extended regular expressions and that which is compatible with the standard SQL LIKE predicate.

This type of pattern may include the following special characters:

- The percentage sign (%) matches zero or more characters.
- The underscore (_) matches exactly one character.
- All other characters are directly matched.
- The special meaning for the percentage sign and the underscore character in the pattern may be overridden by preceding these characters with an escape character, which is the pound sign (#) in this implementation.

With respect to the extended regular expression matching syntax (**=~**), anomalous behavior may result if the locale of the RMC daemon (for event expression evaluation) or the Resource Manager daemons (for select string evaluation) is not a UTF-8 locale. Expressions and select strings passed into the RMC subsystem are converted to UTF-8. The **=~** operator is implemented via **regcomp()** and **regexec()**, which are affected by the locale. Therefore, if the locale is not UTF-8, unexpected results may occur.

Qualifiers

A mechanism is needed that permits an RMC client to qualify an event. In other words, an event may be of some interest, but only if some other condition is also met.

Normally, an expression that evaluates to True results in the generation of an event. But, it might be the case that a single event is not of much interest. Consider the "file system close to full" example:

```
PercentTotUsed > 90
```

While it is interesting that a file system is almost full, to a system administrator responsible for managing file systems, what might be of more interest is that a file system is close to full and remains in that condition for a period of time. A temporary spike in usage can be ignored.

A *qualifier* is an extension to the expression syntax described in the preceding topics in this section ("Using expressions to specify condition events and command selection strings" on page 196) that specifies this other condition. A qualifer consists of a double underscore (__), the string **QUAL**, a single underscore (_), and the qualifer name (*xxxx*). It has the following general form:

```
expression __QUAL_xxxx(arg1, arg2, ...)
```

A qualifier is appended to the end of an expression, separated by one or more blanks. A qualifier can be used with a primary expression or a re-arm expression.

The **__QUAL_COUNT(***arg1***, *arg2***)** qualifier counts the number of True expression evaluations. Once the *arg1* number of True evaluations have occurred, the event notification is generated. However, this count is maintained within a moving window of the last *arg2* consecutive evaluations. Once *arg2* consecutive evaluations have occurred, prior to performing the next evaluation, the count of True evaluations is reduced by one if the oldest evaluation in the window was True. When an event notification is generated, the count of True evaluations and consecutive evaluations is set to **0**.

The value for *arg1* must be less than or equal to the value for *arg2*. Continuing with the file system full example, consider the following primary and re-arm expressions, respectively:

```
PercentTotUsed > 90 __QUAL_COUNT(7, 10)
PercentTotUsed < 60
```

If seven out of the last 10 primary expression evaluations are True, an event notification is generated and the re-arm expression is evaluated until it is True. In simpler terms, if seven out of the last 10 samples of file system usage were greater than 90%, an event is generated. Another event will not be generated until the file system usage drops below 60%.

If all the attributes in the primary expression have a regular period, *arg2* can be considered a duration over which the count of True evaluations is maintained. The actual duration is a function of the shortest and longest reporting interval associated with the attributes specified in the expression, as given by:

```
min_interval * arg2 <= duration <= max_interval * arg2
```

For this example, the duration is 10 minutes.

The **__QUAL_RATE(***arg1***, *arg2***)** qualifier specifies a rate of True expression evaluations that must be achieved before an event notification is generated. *arg1* is a count of True evaluations and *arg2* is a number of seconds. An event notification is generated when the last *arg1* True evaluations have occurred within *arg2* seconds. Once *arg1* True evaluations have occurred, prior to performing the next evaluation, the count of True evaluations is reduced by one. False evaluations are ignored and not counted. When an event notification is generated, the count of True evaluations is set to **0**. Note that the rate calculation is not performed until *arg1* True evaluations have occurred. The time at which this occurs is a function of the time a new value is reported for any of the attributes in the expression.

The rate qualifier is probably more appropriate for expressions containing attributes of variable type **State**, that is, the periodic evaluation is irregular. If the operational state of a resource is expected to change, but not change rapidly, this expression is useful:

```
OpState != OpState@P __QUAL_RATE(5, 60)
```

If five True evaluations occur within one minute, an event is generated. Note that this qualifier is not an instantaneous rate calculation, that is, an event is not generated if the express is True every (*arg1 / arg2*) seconds, or every 0.0833 seconds in the preceding example.

Qualifiers can be used with re-arm expressions as well.

Custom dynamic attributes

The RMC subsystem supports the definition of custom dynamic attributes within a resource class and their use in event expressions.

Custom dynamic attributes are run-time extensions to the resource dynamic attributes that are returned by the **lsrsrcdef -ad** *resource_class_name* command. Custom dynamic attributes are defined for each resource. The manner in which custom dynamic attributes are defined is a function of the resource class implementation. Custom dynamic attributes cannot be defined as class dynamic attributes.

Custom dynamic attributes are supported if the resource class defines the **CustomDynamicAttributes** persistent resource attribute. This persistent attribute is of type SD Array and contains one array element for each custom dynamic attribute that may be defined for the resource. For any given resource, if this persistent attribute has an array value with zero elements, custom dynamic attribute are not defined for the resource. This SD contains the following fields:

- **Name**, which is the name of the custom dynamic attribute.
- **ID**, which is the ID of the custom dynamic attribute.
- **DataType**, which is the datatype of the custom dynamic attribute.

  Custom dynamic attributes cannot be a Structured Data type.
- **VariableType**, which is the variable type of the custom dynamic attribute: **Counter**, **Quantity**, **Quantum**, or **State**. For more information about variable types, see "Dynamic attributes" on page 107.
- **ReportingInterval**, which is the reporting interval of the custom dynamic attribute.
- **Properties**, which are the properties of the custom dynamic attribute.

  One property is supported. If Bit 0 of the property value is set, the attribute must be monitored using an event expression in order to query its value using the **lsrsrc** command.

Each resource does not need to define the same custom dynamic attributes. However, when using the **mkcondition** command, all selected resource must have matching values for their **CustomDynamicAttributes** persistent resource attributes if any custom dynamic attributes are used in the event expressions. If the RMC subsystem cannot obtain the values of the **CustomDynamicAttributes** persistent resource attributes for the selected resources, the **mkcondition** command returns an error in its response, indicating that the custom dynamic attributes in the expressions could not be validated.

Examples of expressions

Expressions can take various forms.

Some examples of the types of expressions that can be constructed follow:

1. The following expressions match all rows or resources that have a name which begins with 'tr' and ends with '0', where 'Name" indicates the column or attribute that is to be used in the evaluation:

   ```
   Name =~'tr.*0'
   Name LIKE 'tr%0'
   ```

2. The following expressions evaluate to TRUE for all rows or resources that contain 1, 3, 5, 6, or 7 in the column or attribute that is called IntList, which is an array:

   ```
   IntList|<{1,3,5..7}
   IntList in (1,3,5..7)
   ```

3. The following expression combines the previous two so that all rows and resources that have a name beginning with 'tr' and ending with '0' and have 1, 3, 5, 6, or 7 in the IntList column or attribute will match:

```
(Name LIKE "tr&(IntList|<(1,3,5..7))
(Name=~'tr.*0') AND (IntList IN {1,3,5..7})
```

Creating an RSCT management domain

A management domain is defined as a set of nodes whose resources can be managed and monitored from one of the nodes that is designated as the management control point (MCP). All other nodes are considered to be managed nodes. A managed node can be configured in multiple management domains at the same time.

The CSM licensed program creates management domains. The CSM management server is the MCP of the domain.

Logical partitions (LPARs) on the Power Systems servers that are installed with the AIX operating system are automatically configured as managed nodes in a management domain in which the Hardware Management Console (HMC) or the Integrated Virtualization Manager (IVM) is the MCP. If the service and productivity tools for Linux on the Power Systems platform are installed, the Linux LPARs are automatically configured as managed nodes in a management domain in which the HMC or IVM is the MCP. If an LPAR is managed by more than one HMC, this LPAR is a managed node in each resulting management domain.

If the resource monitoring and control (RMC) monitoring option for xCAT is selected, you can use xCAT commands to create a management domain. In the management domain, the service node is the MCP and the nodes that the service node manages are the managed nodes of the management domain.

You can manually create a management domain if you are not using CSM or xCAT. If CSM is installed on a POWER LPAR that is to be configured as a managed node, the CSM requirements must be as follows:

- On an AIX LPAR, **csm.client** 1.7.1.4 (PTF U834202) must be installed.
- On a Linux LPAR, CSM 1.7.1.6, or later must be installed.

The nodes that are defined in one or more peer domains can also be configured as managed nodes in a single management domain. Also, management domains can be hierarchical. That is, a managed node in one management domain can be designated as the MCP of another management domain.

To create and modify a management domain, use the RMC commands: **mkrsrc**, **chrsrc**, and **rmrsrc**.

Prerequisites and restrictions when manually creating a management domain

There are various prerequisites and restrictions when you manually create a management domain.

Before you use RMC commands, be aware of the following prerequisites and restrictions:

- On the AIX operating system, the necessary RSCT filesets are installed automatically.
- On the Linux operating system, the `rsct.core`, `rsct.core.utils`, and `src` packages (RPMs) are required.
- All nodes that you plan to include in the management domain must be reachable from the MCP. Although you can have multiple networks and routers to accomplish this requirement, there must be IP connectivity between the MCP and each managed node.
- The network interfaces that are associated with the configured IP addresses in the management domain, and are in the same subnet, must be configured to use the same maximum transmission unit (MTU) size. It is important because MTU size differences between interfaces in the same subnet can result in packet loss.
- The MCP cannot be defined in the same peer domain as any of its managed nodes.
- If a managed node is in a peer domain, all nodes in that peer domain must be managed nodes in the same management domain.

- Circular management domains are not permitted. That is, if node_A is designated as an MCP and node_B is one of its managed nodes, node_A cannot be defined as a managed node to node_B when node_B is also designated as an MCP.
- In case of management domain on the Linux operating system for PowerPC® (PPC) architecture, the managed nodes require the 32-bit platform enablement library (the `librtas` package) to access certain functions that are provided by the operating system firmware. Therefore, you must ensure that the 32-bit `librtas` package is installed on all the managed nodes. If it is not installed, install the 32-bit `librtas` package from the installation media.

Creating the domain

To create a management domain, use the **mkrsrc** command to create **IBM.MngNode** resources on the designated MCP and to create an **IBM.MCP** resource on each managed node.

Select the node that is to be the MCP and select the nodes that are to be managed nodes. On the MCP, determine the IP addresses to be used by RMC on the managed nodes to communicate with RMC on the MCP. On each managed node, determine the IP addresses to be used by RMC on the MCP to communicate with RMC on the managed node. If multiple IP addresses are available on a node, one or more can be used for RMC communication. If more than one IP address of a node is used, these addresses must be assigned to network interfaces on separate subnets.

On each of these nodes, obtain the RSCT node ID. The node ID can be found in the `/etc/ct_node_id` or `/var/ct/cfg/ct_node_id` files. The contents of these files are identical. The first line contains the node ID, which is a hexadecimal value.

To enable remote RMC client connections, run this command on each of these nodes:

```
rmcctrl -p
```

On the selected MCP, an **IBM.MngNode** resource is created to configure the managed node to the RMC subsystem on the MCP. After this resource is created, an **IBM.MCP** resource is created on the corresponding managed node to configure the selected MCP to the RMC subsystem on the managed node. The **IBM.MngNode** resource must be created on the MCP before the **IBM.MCP** resource is created on the managed node. Presumably, there are a number of managed nodes. You can create all of the **IBM.MngNode** resources with a single execution of the **mkrsrc** command. Then, create the **IBM.MCP** resources on the managed nodes.

For each **IBM.MngNode** resource, you must specify these attributes: **`IPAddresses`**, **`KeyToken`**, **`Name`**, and **`NodeID`**. Optionally, you can also specify the **`Aliases`** attribute. The following list describes the **IBM.MngNode** attributes:

Aliases
: Specifies one or more names of the managed node. These names can be any meaningful unique identifiers. Any of these names can be used in selection strings as the value of the **`NodeNameList`** attribute instead of the value of the **`Name`** attribute.

IPAddresses
: Specifies a list of IP addresses assigned to the managed node. Each address must be on a separate subnet.

KeyToken
: Specifies the host identity that is used by the RMC subsystem to look up the public key of the managed node in the trusted host list on the MCP. This value is typically the fully qualified host name or IP address of the managed node.

Name
: Typically, identifies the primary host name of the managed node. However, any unique name can be used. This name is used in selection strings to identify resources on the managed node, as the value of the **`NodeNameList`** attribute.

NodeID
Identifies the node ID of the managed node. The node ID that is obtained earlier is used. The characters `0x` must be prefixed to the ID.

For each **IBM.MCP** resource, you must specify these attributes: **IPAddresses**, **KeyToken**, **MNName**, and **NodeID**. The following list describes the **IBM.MCP** attributes:

IPAddresses
Specifies a list of IP addresses assigned to the MCP. Each address must be on a separate subnet.

KeyToken
Specifies the host identity that is used by the RMC subsystem to look up the public key of the MCP in the trusted host list on the managed node. This value is typically the fully qualified host name or IP address of the MCP.

MNName
Identifies the name of the managed node as known to the MCP. This attribute is typically the value of the **Name** attribute of the corresponding **IBM.MngNode** resource. This name is returned as the value of the **NodeNameList** attribute for managed node resources that are selected by RMC commands on the MCP.

NodeID
Identifies the node ID of the MCP. The node ID obtained earlier is used. The characters `0x` must be prefixed to the ID.

As each **IBM.MCP** resource is created on a managed node, the public key exchange protocol is executed. This protocol depends on the corresponding **IBM.MngNode** resource to be created. This protocol exchanges the public keys of the MCP and the managed node. It also creates authorization for the root user on the MCP to have read and write access to all of the resources on the managed node. Any other user on the MCP has read access to all of the resources on the managed node.

This protocol can be started manually by running the following command on the managed node at any time:

```
refrsrc IBM.MCP
```

The protocol is run with all MCPs for which there are **IBM.MCP** resources created.

Example:

On the selected MCP, create a file named `infile.mngnodes` with the following contents, replacing attribute values with your system values:

```
PersistentResourceAttributes::
Resource 1:
    Name = "e96m5fsp02"
    NodeID = 0x8e5e73b6db758deb
    KeyToken = "e96m5fsp02.ppd.pok.ibm.com"
    IPAddresses = {"9.114.42.216"}
Resource 2:
    Name = "e96m5fsp04"
    NodeID = 0xd8e61be41aa6b35f
    Aliases = {"mike","joe"}
    KeyToken = "e96m5fsp04.ppd.pok.ibm.com"
    IPAddresses = {"9.114.42.218"}
```

Then, run this command:

```
mkrsrc -f infile.mngnodes IBM.MngNode
```

On the managed node named `e96m5fsp02`, create a file named `infile.mcp` with the following contents, replacing attribute values with those for your system:

```
PersistentResourceAttributes::
Resource 1:
        MNName = "e96m5fsp02.ppd.pok.ibm.com"
        NodeID = 0xf53a576a3e76a966
        KeyToken = "e96m5fsp05.ppd.pok.ibm.com"
        IPAddresses = {"9.114.42.219"}
```

Then, run this command:

```
mkrsrc -f infile.mcp IBM.MCP
```

On the managed node named `e96m5fsp04`, create a file named `infile.mcp` with the following contents, replacing attribute values with those for your system:

```
PersistentResourceAttributes::
Resource 1:
        MNName = "e96m5fsp04.ppd.pok.ibm.com"
        NodeID = 0xf53a576a3e76a966
        KeyToken = "e96m5fsp05.pok.ibm.com"
        IPAddresses = {"9.114.42.219"}
```

Then, run this command:

```
mkrsrc -f infile.mcp IBM.MCP
```

Result:

On the MCP, run the **lsrsrc** command:

```
lsrsrc IBM.MngNode Name NodeID KeyToken Aliases IPAddresses Status
```

The following output is displayed:

```
Resource Persistent and Dynamic Attributes for IBM.MngNode
resource 1:
        Name        = "e96m5fsp04"
        NodeID      = 15629210223349511007
        KeyToken    = "e96m5fsp04.ppd.pok.ibm.com"
        Aliases     = {"mike","joe"}
        IPAddresses = {"9.114.42.218"}
        Status      = 1
resource 2:
        Name        = "e96m5fsp02"
        NodeID      = 10258764230399725035
        KeyToken    = "9.114.42.216"
        Aliases     = {}
        IPAddresses = {"9.114.42.216"}
        Status      = 1
```

The node ID is displayed in decimal format. The value of the **Status** attribute indicates that the RMC subsystem on the managed node is communicating with the RMC subsystem on the MCP. This value is 0 if the subsystems are not communicating.

On a managed node, run the **lsrsrc** command:

```
lsrsrc IBM.MCP MNName NodeID KeyToken IPAddresses Status
```

The following output is displayed:

```
Resource Persistent and Dynamic Attributes for IBM.MCP
resource 1:
        MNName      = "e96m5fsp02.ppd.pok.ibm.com"
        NodeID      = 17670532201767676262
        KeyToken    = "e96m5fsp05.ppd.pok.ibm.com"
        IPAddresses = {"9.114.42.219"}
        Status      = 1
```

The node ID is displayed in decimal format. The value of the **Status** attribute indicates that the RMC subsystem on the MCP is communicating with the RMC subsystem on the managed node. This value is 0 if the subsystems are not communicating.

Modifying the domain

You can modify the management domain as the need arises, by simply adding or removing **IBM.MngNode** resources. A managed node can be added to another domain by adding an **IBM.MCP** resource or removed from a domain by removing an **IBM.MCP** resource. Use the **mkrsrc** command to add a resource. Use the **rmrsrc** command to remove a resource. You can modify certain attributes of the resources by using the **chrsrc** command.

When you add new nodes, collect node ID and IP address information as described under **Creating the domain**. Run the **rmcctrl -p** command on new nodes.

For LPARs on the Power Systems servers, **IBM.MCP** resources are created automatically for each HMC or IVM that is managing the LPAR. These resources have additional attributes, as shown in this example:

```
# lsrsrc IBM.MCP
Resource Persistent Attributes for IBM.MCP
resource 1:
        MNName            = "9.114.42.216"
        NodeID            = 13115103585919069835
        KeyToken          = "e96m5fsp01.ppd.pok.ibm.com"
        IPAddresses       = {"9.114.47.81"}
        ConnectivityNames = {"9.114.42.216"}
        HMCName           = "9133-55A*10B7D1G"
        HMCIPAddr         = "9.114.47.81"
        HMCAddIPs         = ""
        HMCAddIPv6s       = ""
        ActivePeerDomain  = ""
        NodeNameList      = {"e96m5fsp02.ppd.pok.ibm.com"}
```

If you find resources that have the **HMCName** attribute that is set to a non-null value, do not remove or modify these resources. Doing so damages the ability of the LPAR to be managed by the HMC or IVM.

To add new managed nodes to the MCP of your management domain, use the **mkrsrc** command to create the new **IBM.MngNode** resources as described in **Creating the domain**. Then, add the **IBM.MCP** resource to each new managed node by using the **mkrsrc** command.

To remove managed nodes from your management domain, use the **lsrsrc** command on the MCP to identify the **IBM.MngNode** resources that represent the managed nodes. Then, using the value of the **Name** attribute or the **NodeID** attribute of the resources you want to remove in a selection string, run the **rmrsrc** command. After the **IBM.MngNode** resources are removed, on each associated managed node use the **lsrsrc** command to identify the **IBM.MCP** resource that represents the MCP. Then, using the value of the **NodeID** attribute of the resource you want to remove in a selection string, run the **rmrsrc** command.

Example:

On the MCP, run the **lsrsrc** command:

```
lsrsrc IBM.MngNode
```

The following output is displayed:

```
Resource Persistent Attributes for IBM.MngNode
resource 1:
        Name                  = "e96m5fsp04"
        NodeID                = 15629210223349511007
        KeyToken              = "e96m5fsp04.ppd.pok.ibm.com"
        Aliases               = {"mike","joe"}
```

```
        IPAddresses            = {"9.114.42.218"}
        HWType                 = ""
        HWModel                = ""
        HWSerialNum            = ""
        LParID                 = ""
        StorageFacilityImageID = ""
        ActivePeerDomain       = ""
        NodeNameList           = {"e96m5fsp05.ppd.pok.ibm.com"}
resource 2:
        Name                   = "9.114.42.216"
        NodeID                 = 10258764230399725035
        KeyToken               = "9.114.42.216"
        Aliases                = {}
        IPAddresses            = {"9.114.42.216"}
        HWType                 = ""
        HWModel                = ""
        HWSerialNum            = ""
        LParID                 = ""
        StorageFacilityImageID = ""
        ActivePeerDomain       = ""
        NodeNameList           = {"e96m5fsp05.ppd.pok.ibm.com"}
```

Assuming that the node identified by `e96m5fsp04` is to be removed, run the following command:

```
rmrsrc -s 'Name == "e96m5fsp04"' IBM.MngNode
```

On the managed node that is identified by `e96m5fsp04`, run the following command:

```
lsrsrc IBM.MCP
```

The following output is displayed:

```
Resource Persistent Attributes for IBM.MCP
resource 1:
        MNName             = "e96m5fsp04.ppd.pok.ibm.com"
        NodeID             = 17670532201767676262
        KeyToken           = "e96m5fsp05.pok.ibm.com"
        IPAddresses        = {"9.114.42.219"}
        ConnectivityNames  = {}
        HMCName            = ""
        HMCIPAddr          = ""
        HMCAddIPs          = ""
        HMCAddIPv6s        = ""
        ActivePeerDomain   = ""
        NodeNameList       = {"e96m5fsp04.ppd.pok.ibm.com"}
resource 2:
        MNName             = "9.114.42.218"
        NodeID             = 13115103585919069835
        KeyToken           = "e96m5fsp01.ppd.pok.ibm.com"
        IPAddresses        = {"9.114.47.81"}
        ConnectivityNames  = {"9.114.42.218"}
        HMCName            = "9133-55A*10B7D1G"
        HMCIPAddr          = "9.114.47.81"
        HMCAddIPs          = ""
        HMCAddIPv6s        = ""
        ActivePeerDomain   = ""
        NodeNameList       = {"e96m5fsp04.ppd.pok.ibm.com"}
```

The resource to be removed is the resource with the **IPAddresses** value of 9.114.42.219. Run the following command:

```
rmrsrc -s 'NodeID == 17670532201767676262' IBM.MCP
```

You can modify these **IBM.MngNode** resource attributes: **Aliases**, **IPAddresses**, **KeyToken**, and **Name**.

You can modify these **IBM.MCP** resource attributes: **IPAddresses**, **KeyToken**, and **MNName**.

The **NodeID** attribute cannot be modified for either type of resource. This attribute uniquely identifies the node to the RMC subsystem. Normally, the RSCT node ID never changes. However, you change the node ID by using the **recfgct** command. If the node ID of a managed node or an MCP is changed, the resource that is identified by the old node ID must be removed and a new resource added. It is only done for resources you created.

If the **KeyToken** attribute is modified on an **IBM.MngNode** resource, the **refrsrc IBM.MCP** command must subsequently be run on the corresponding managed node. Likewise, this command must be run whenever the **KeyToken** attribute is modified on an **IBM.MCP** resource.

Example:

To modify the **Aliases** attribute of an **IBM.MngNode** resource on the MCP, identify the resource by using the **lsrsrc** command:

```
lsrsrc IBM.MngNode
```

The following output is displayed:

```
# lsrsrc IBM.MngNode
Resource Persistent Attributes for IBM.MngNode
resource 1:
        Name                   = "e96m5fsp04"
        NodeID                 = 15629210223349511007
        KeyToken               = "e96m5fsp04.ppd.pok.ibm.com"
        Aliases                = {"mike","joe"}
        IPAddresses            = {"9.114.42.218"}
        HWType                 = ""
        HWModel                = ""
        HWSerialNum            = ""
        LParID                 = ""
        StorageFacilityImageID = ""
        ActivePeerDomain       = ""
        NodeNameList           = {"e96m5fsp05.ppd.pok.ibm.com"}
resource 2:
        Name                   = "9.114.42.216"
        NodeID                 = 10258764230399725035
        KeyToken               = "9.114.42.216"
        Aliases                = {}
        IPAddresses            = {"9.114.42.216"}
        HWType                 = ""
        HWModel                = ""
        HWSerialNum            = ""
        LParID                 = ""
        StorageFacilityImageID = ""
        ActivePeerDomain       = ""
        NodeNameList           = {"e96m5fsp05.ppd.pok.ibm.com"}
```

If you want to replace the alias joe and use the alias john instead, run the **chrsrc -s** command:

```
chrsrc -s 'NodeID == 15629210223349511007' IBM.MngNode Aliases='{"mike","john"}'
```

Result:

Run the **lsrsrc -s** command to verify the change:

```
lsrsrc -s 'NodeID == 15629210223349511007' IBM.MngNode
```

The following output is displayed:

```
# lsrsrc -s 'NodeID == 15629210223349511007' IBM.MngNode
Resource Persistent Attributes for IBM.MngNode
resource 1:
        Name                   = "e96m5fsp04"
        NodeID                 = 15629210223349511007
```

```
KeyToken               = "e96m5fsp04.ppd.pok.ibm.com"
Aliases                = {"mike","john"}
IPAddresses            = {"9.114.42.218"}
HWType                 = ""
HWModel                = ""
HWSerialNum            = ""
LParID                 = ""
StorageFacilityImageID = ""
ActivePeerDomain       = ""
```

Related information:

"Configuring cluster security services" on page 267
You can administer cluster security services for management domains and RSCT peer domains.

mkrsrc command

chrsrc command

rmrsrc command

Controlling access to root commands and scripts

The RSCT least-privilege (LP) resource manager is a client-server application that allows you to enhance the security, performance, and control of applications that require root authority to run.

The LP resource manager, or LPRM, runs on both AIX and Linux nodes. Through the LP resource manager, you can:

- Define specific root commands or scripts as LP resources. An LP resource represents a least-privilege access command or script. Least-privilege capability allows a select group of authorized users to run the command or script without needing complete root authority.
- Enable distributed and parallel execution of these LP resources. Authorized users can run the command or script locally or remotely, on one or simultaneously on many nodes, without having to log into each node on which the command or script is to run.
- Monitor and manage LP resources and operations on one node or across many nodes. The LP resource manager uses the Audit log resource manager to log detailed usage information about LP resources.

Overview of LP resource manager operation

The essential service provided by the LP resource manager is to allow specific non-root users to be authorized to run specific commands requiring root authority.

For example, a non-**root** user on a management server would not normally be able to power off a managed node using the **rpower** command. However, the root user could give a user, **user01**, for example, authority to do so by defining an LP resource with the following command:

```
[root@ms-node]# mklpcmd rpower /opt/csm/bin/rpower user01@LOCALHOST rx
```

Once this LP resource was defined, the user **user01** could power off managed node *nodeA* by running the **rpower** command through the LP resource manager, using the following command:

```
[user01@ms-node]$ runlpcmd rpower -n nodeA off
```

The LP resource manager consists of two parts, a client program and a daemon. Instances of both the client and daemon run on each node. The nodes can be independent workstations or they can be in a management domain or a peer domain.

The LP resource manager provides one resource class, **IBM.LPCommands**, that represents root commands or scripts. Through this representation of resources, the LP resource manager can run a root command or script, locally or remotely, on behalf of an authorized user. When the resource's processing completes, the LP resource manager returns the processing results to the user. More specifically, the resource manager:

- Allows administrators to manage LP resources by defining, changing, and removing them. In addition to resource monitoring and control (RMC) commands, administrators can use the following LPRM commands to manage LP resources:
  chlpcmd
  Changes certain attributes of an LP resource.
  lphistory
  Lists a particular number of previously issued LPRM commands.
  lslpcmd
  Lists one or more LP resources on one or more nodes in a domain.
  mklpcmd
  Defines an LP resource to the RMC subsystem.
  rmlpcmd
  Removes an LP resource from one or more nodes in a domain.
  runlpcmd
  Runs a particular LP resource on one or more nodes in a domain.
- Enables local or remote execution of the LP resources from one or more nodes within a management or peer domain. Two environment variables, CT_CONTACT and CT_MANAGEMENT_SCOPE, affect which LPRM daemon runs and its scope of operation. Further details appear in "Determining the target nodes for an LPRM command" on page 224.
- Secures access to the root commands or scripts by using a set of access control lists (ACLs) that can be modified using LP resource manager commands. For more information, see "Overview of the LP resource manager's access control lists."

Overview of the LP resource manager's access control lists

Although authorization for most resource classes and resources is determined by settings in an RMC ACL file, this is not true for the IBM.LPCommands resource class or resources of the **IBM.LPCommands** resource class.

Instead, the LP resource manager has its own set of access control lists (LP ACLs) that provide a finer level of authorization for LP resources. Specifically, the LP ACLs enable you to specify permissions for a particular resource (while the RMC ACL only allows user permission to be specified for a resource class or all resources of a resource class).

Note: Although in RSCT versions prior to 2.4.2.0, permissions for the **IBM.LPCommands** resource class were specified in an RMC ACL file, this is no longer allowed. An entry for the **IBM.LPCommands** resource class in an RMC ACL file is ignored.

Related information:

"Managing user access to resources using RMC ACL files" on page 126
RMC implements authorization using an *access control list* (ACL) file. Specifically, RMC uses the ACL file on a particular node to determine the permissions that a user must have in order to access resource classes and their resource instances.

Types of LP ACLs

There are four types of LP ACLs that control access to the IBM.LPCommands resource class and its resources.

To understand the roles played by the LP ACL types, it is necessary to understand that basic operations within the RMC subsystem fall into two categories — operations that are applied to a class, and operations that are applied to a resource. Access to IBM.LPCommands class operations is controlled by an ACL associated with the class. Access to IBM.LPCommands resource operations is controlled by an ACL associated with the resource.

The ACL type that controls access to class operations is known as a Class ACL. The primary ACL type that controls access to resource operations is known as a Resource ACL. There are two secondary types of ACLs related to resources — Resource Initial ACLs and Resource Shared ACLs.

An LPRM administrator performs administrative tasks using LPRM commands. Some LPRM commands only perform class operations. Access control for those commands is performed by a Class ACL. Other LPRM commands perform both class and resource operations. Access control for those commands is performed by a Class ACL and a Resource ACL. In a way, this is analogous to how access to files is controlled on a traditional UNIX filesystem. Access to some file operations, such as creating a file, is controlled solely by permissions on directories. Access to other file operations, such as reading a file, is controlled by permissions on both directories and the file itself.

Table 62 provides more details concerning the four types of LP ACLs.

Table 62. The four types of LP ACLs

ACL type	Description	Command to list the ACL entries	Command to modify the ACL
Class ACL	A list of entries that define access permissions for operations on the IBM.LPCommands resource class. Some LPRM commands only involve operations on the IBM.LPCommands resource class. For those commands, a user will need to have the correct access permissions specified in a Class ACL. The commands that define a LP resource, remove a LP resource, and list the history of commands executed through LP resources fall into this category: • **mklpcmd** • **rmlpcmd** • **lphistory** Other LPRM commands involve operations on the IBM.LPCommands resource class and on a resource itself. For those commands, a user will need to have the correct access permissions specified in a Class ACL and a Resource ACL (see description below). The commands that list a LP resource, change a LP resource, and execute a command through a LP resource fall into this category: • **lslpcmd** • **chlpcmd** • **runlpcmd** There is one Class ACL per node.	**lslpclacl**	**chlpclacl**
Resource ACL	A list of entries that define access permissions for operations on a particular resource of the IBM.LPCommands resource class. A user will need to have the correct access permissions specified in this ACL to change LP resources using the **chlpcmd** command, or list LP resources using the **lslpcmd** command. A user will need to have the correct access permissions specified in this ACL to run a root command or script (IBM.LPCommands resource) using the **runlpcmd** command. When the **mklpcmd** command is run to create a new root command or script, a Resource ACL is created for the new IBM.LPCommands resource. The entries in this Resource ACL are copied from the Resource Initial ACL, and then modified by the ACL entries optionally specified with the **mklpcmd** command. A Resource ACL can specify that the Resource Shared ACL (see description below) should be used instead to control access to this resource. There is one Resource ACL per LP resource.	**lslpracl**	**chlpracl**
Resource Initial ACL	A list of default entries that will be copied to a new Resource ACL when the **mklpcmd** command is issued to create a new root command or script (IBM.LPCommands resource). The new Resource ACL is then modified by the ACL entries optionally specified with the **mklpcmd** command. There is one Resource Initial ACL per node.	**lslpriacl**	**chlpriacl**

Table 62. The four types of LP ACLs (continued)

ACL type	Description	Command to list the ACL entries	Command to modify the ACL
Resource Shared ACL	A list of entries that define access permissions that can be applied to multiple resources. An individual Resource ACL can specify (using the **chlpracl** command) that it should be bypassed and that the Resource Shared ACL should instead be used to control access to the resource. If a Resource ACL is bypassed in favor of the Resource Shared ACL, then the user will need to have the correct access permissions specified in the Resource Shared ACL to run commands that perform resource operations, such as **chlpcmd**, **lslpcmd**, and **runlpcmd**. The ability to bypass one or more individual Resource ACLs in favor of the Resource Shared ACL enables you to apply one set of access controls to multiple resources. There is one Resource Shared ACL per node.	**lslprsacl**	**chlprsacl**

All of the LP commands described here perform operations on the IBM.LPCommands resource class. When a user issues a command that operates on the IBM.LPCommands resource class, RMC will check the Class ACL to verify that the user has the correct permission to perform the operation. If not, the operation is rejected and the command returns an error.

Some of the LP commands (such as **chlpcmd**, **lslpcmd**, and **runlpcmd**) also perform an operation on a resource of the IBM.LPCommands resource class. When a user issues a command that operates on a particular resource of the IBM.LPCommands resource class, RMC first checks the resource's individual Resource ACL to determine if the Resource ACL contains an access list or if the Resource Shared ACL is instead used to protect the resource.

- If the Resource ACL for the resource has an access list, then RMC checks the Resource ACL to verify that the user has the correct permission to perform the operation.
- If the Resource ACL for the resource indicates that the Resource Shared ACL should be used for the resource, then RMC checks the Resource Shared ACL to verify that the user has the correct permission to perform the operation.

If the user does not have the correct permission, the operation is rejected and the command results in an error.

Format of an ACL entry

Regardless of the type of LP ACL, the format of ACL entries is the same. An ACL entry consists of a user identifier and an associated set of user permissions.

The user identifier in an ACL entry:

The user identifier can take one of several forms.

The user identifier can take one of the forms described in Table 63 on page 218.

Table 63. LP ACL user identifier forms

This user identifier form...	Specifies...
host:*host_user_identifier*	A host user identifier. The **host:** keyword is optional. It specifies that the user identifier can be matched against a network identifier provided by the host based authentication (HBA) security mechanism. If the **host:** keyword is omitted and the entry does not take one of the other forms outlined in this table, the entry is assumed to be a host user identifier. The host user identifier can take a number of different formats: *user_name@host_identifier* Specifies a particular user. The *host_identifier* portion of this specification can take a number of forms. These forms are the same as when the host user identifier format is specified as a *host_identifier* alone. *host_identifier* Specifies any user running the RMC application on the host identified. The *host_identifier* can be: • A fully qualified host name • A short host name • An IP address • An RSCT node ID. This is a 16-digit hexadecimal number. For example, 0xaf58d41372c47686. • The keyword LOCALHOST. This keyword is a convenient way to specify the RSCT node ID of the node where the ACL exists. * Specifies any user running an RMC application on any host.
none:*mapped_user_identifier*	A mapped name as specified in the **ctsec_map.global** or **ctsec_map.local** file. See "Configuring the host based authentication (HBA) mechanism mappings" on page 281 for more information on creating these mapped names.
UNAUTHENT	An unauthenticated user.

When specifying a host based authentication (HBA) security mechanism ACL entry, whether it is appropriate to use a fully qualified host name, a short host name, an IP address, or an RSCT node ID to identify the host depends on how the LPRM commands whose access are to be controlled by the ACL will connect to the RMC subsystem. For more information, see "Specifying host based authentication ACL entries" on page 232.

When using an LPRM command to add an ACL entry to an ACL, it is often necessary to specify an RSCT node ID for the HBA *host_identifier*. The RSCT node ID can be specified using the LOCALHOST keyword, the NODEID keyword, or the actual 16-digit hexadecimal number, as follows:

- When the desired node ID is that of the node on which the ACL exists, the LOCALHOST keyword can be specified. This keyword is stored in the ACL itself. When the ACL is listed, the LOCALHOST keyword will be shown.
- When the desired node ID is that of the node on which the ACL editing command is being run, the NODEID keyword can be specified. This keyword is not stored in the ACL itself. Rather, the ACL editing command looks up the node ID of the node on which the command is being run and places that node ID in the ACL entry.
  - When an ACL editing command is run on the same node on which the ACL exists, the node ID represented by LOCALHOST and NODEID are the same. In this case, the LOCALHOST keyword may be preferable because it is actually stored in the ACL and may be easier to read than the 16-digit hexadecimal number.
  - When an ACL editing command is run on one node to remotely edit an ACL on another node, the node ID represented by LOCALHOST and NODEID are different. Which keyword to use, if any,

depends on what node ID must be placed in the ACL entry — the node ID of the node on which the ACL editing command is being run (NODEID) or the node ID of the node on which the ACL exists (LOCALHOST).

- If the desired node ID is different than those represented by the LOCALHOST and NODEID keywords, then the actual 16-digit hexadecimal number must be specified.

There are several ways to determine the node ID of a node. For example, if neither the CT_CONTACT nor CT_MANAGEMENT_SCOPE environment variables are set, you can use the **lsrsrc** command to determine the node ID of the node on which the command is run:

```
lsrsrc IBM.Host NodeIDs
```

To determine the node IDs for all nodes in a management domain or an RSCT peer domain, use this command:

```
lsrsrc -ta IBM.Host NodeIDs NodeNameList
```

The node IDs returned by the **lsrsrc** command are displayed in decimal. You will need to use the hexadecimal equivalent (prefixed by 0x) when specifying the host identifier. If the nodes are in an RSCT peer domain, you can obtain the hexadecimal version of the node ID using the **lsrpnode** command. To do this, issue the following command from a node that is online in the peer domain:

```
lsrpnode -i
```

User permissions in an ACL entry:

The user permissions define the level of access that a user has to the class or to the resources.

The user permissions are expressed as a string of one or more characters, each representing a particular permission granted to the user specified by the user identifier. These permissions are described in Table 64.

Table 64. User permissions in an LP ACL entry

Specifying this...	Indicates that the specified user at the specified host has...
x	Execute permission. This allows the user to run the root command or script (IBM.LPCommands resource).
a	Administration permission. This allows the user to make changes to an ACL.
r	Read permission. This allows the users to register and unregister events, query attribute values, and validate resource handles. The **r** permission is a composite permission that is composed of the following permissions. While you can, instead of specifying the **r** permission, specify a subset of the following permissions, it prevents the user from performing some operations. The **r** permission is a convenient way of specifying all of the following flags: **q** Indicates that the specified user at the specified host has query permission. This allows the user to query persistent or dynamic attributes. **l** Indicates that the specified user at the specified host has list permission. This allows the user to list resources. **e** Indicates that the specified user at the specified host has event permission. This allows the user to register, query, and unregister events. **v** Indicates that the specified user at the specified host has validate permission. This allows the user to validate resource handles.

Table 64. User permissions in an LP ACL entry (continued)

Specifying this...	Indicates that the specified user at the specified host has...
w	Write permission. This allows the user to run all other command interfaces. The **w** permission is a composite permission that is composed of the following permissions. While you could, instead of specifying the **w** permission, specify a subset of the following permissions, this would prevent the user from performing some operations. The **w** permission is a convenient way to specify all of the following: **d** Indicates that the specified user at the specified host has define permission. This allows the user to define and undefine resources. **c** Indicates that the specified user at the specified host has refresh permission. This allows the user to refresh resource configuration. **s** Indicates that the specified user at the specified host has set permission. This allows the user to set attributes. **o** Indicates that the specified user at the specified host has online permission. This allows the user to bring resources online and take resources offline.
rw	Read and write permission.

Wider implications of LP ACL permissions

- When granting users permission in LP ACLs, it is important to keep in mind the wider implications of doing so. Granting users read (**r**) permission to the LP class and resources is benign. The users will be able to list resource definitions and ACLs. Granting users write (**w**) and administration (**a**) permissions have much wider implications.
- A user granted write permission to an LP resource is able to change any attributes of the resource. By changing the values of the CommandPath, FilterArg, or FilterScript attributes, the user is able to change what users who execute the resource can run through LPRM. Since the CommandPath attribute can be set to the path name of any command, and since LPRM runs commands with root authority, the user can use the LP resource to run any command of his or her choosing, if the user also has execute (**x**) permission to the resource.
- A user granted write permission to a node's LP Class ACL is able to define new LP resources. The same cautions that apply to granting write access to a LP resource apply here.
- A user granted administration permission to the Class ACL can change the Class ACL, the Resource Initial ACL, and the Resource Shared ACL. A user granted administration permission to a Resource ACL can change the ACL.
- Changes to a Resource Initial ACL affect how the Resource ACLs of new resources are initialized. Changes to a Resource Shared ACL potentially affect access to multiple resources.

For more information on the LP ACLs, see the **lpacl** man page or the **lpacl** entry in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Permissions for LPRM commands

Permissions are required to run the LP resource manager commands.

Table 65 on page 221 outlines the permissions that are required to run the LPRM commands.

Table 65. Permissions required to run the LP resource manager commands

LPRM command	Class ACL	Resource ACL	Resource Initial ACL
mklpcmd (no ACL entries specified)	w		
mklpcmd (with ACL entries specified)	rw		ra
chlpcmd	r	w	
rmlpcmd	rw		
lslpcmd	r	r	
lphistory	w		
runlpcmd	r	x	
lslpclacl	r		
lslpriacl	r		
lslprsacl	r		
lslpracl	r	r	
chlpclacl	ra		
chlpriacl	ra		
chlprsacl	ra		
chlpracl	r	ra	

For more information on the LPRM commands, see the man pages or to *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Understanding how the LPRM selects an ACL entry from an ACL

When an LP command is run, execution of that command can involve performing several class and resource operations. Before each operation is performed, the LPRM consults the appropriate ACL to determine if the user has permission to perform the operation. The ACL consulted depends on the type of operation, as discussed previously.

When the access list of an ACL is used to determine if a user has permission to perform an operation, one ACL entry is selected. The permission granted by the selected ACL entry determines if the user is permitted to perform the operation. Let's consider how the ACL entry is selected by the LPRM.

While considering how the LPRM selects an ACL entry, keep in mind that an authenticated RMC client has a network identity, and may have a mapped identity.

The network identity is the security-mechanism specific way of identifying the user. For example, `host:user01@nodeA.network` is the host based authentication (HBA) security mechanism network identity for user *user01* on the node with host name *nodeA.network*, when the user has connected remotely from *nodeA* to the RMC daemon on some other node.

The authenticated RMC client may have a mapped identity. A mapped identity is a mapping from a network identity to a local user name. For example, `none:root` represents the mapped root user in ACL entries. See “Understanding native identity mapping” on page 11 for more information.

Also keep in mind that, in some circumstances, a client may be unauthenticated. If the client is unauthenticated, ACL entry selection is simple. If the UNAUTHENT ACL entry exists, that entry's permissions determine whether the client can perform the operation. If the UNAUTHENT ACL entry does not exist, permission to perform the operation is denied.

To select an ACL entry for an authenticated client, the LPRM:

1. Looks for an ACL entry that exactly matches the client's network identity. If such an ACL entry is found, the permissions in that entry determine whether the user has permission to perform the operation. If no such ACL entry is found, proceed to the next step.

2. Looks through the wildcarded ACL entries for the security mechanism through which the client has been authenticated. The entries are searched in order of appearance in the ACL. The first match is used. Wildcarded ACL entries for the HBA security mechanism specify a host (with no user) or specify all hosts ("*").

   If such an ACL entry is found, the permissions in that entry determine whether the user has permission to perform the operation.

   If no such ACL entry is found, proceed to the next step.
3. If the client has a mapped identity, looks for an ACL entry that exactly matches the client's mapped identity.

   If such an ACL entry is found, the permissions in that entry determine whether the user has permission to perform the operation.

   If no such ACL entry is found, permission to perform the operation is denied.

If the above procedure used by the LPRM does not result in the selection of an ACL entry, permission to perform the operation is denied.

Example: The following commands show the Resource ACL for LP resource *rootcmd* on node *ms_node*.

```
[root@ms_node]# lslpracl rootcmdResource ACLs for LPRM
Name    Identity              Permissions NodeName
rootcmd host:root@LOCALHOST   rwa         ms_node.network
rootcmd host:user01@LOCALHOST rx          ms_node.network
rootcmd host:user03@LOCALHOST 0           ms_node.network
rootcmd host:LOCALHOST        r           ms_node.network
```

Now we will consider what happens when three users attempt to list and execute the LP resource *rootcmd*.

First, the user *user01* tries it.

```
[user01@ms_node]# lslpcmd rootcmdName = rootcmd
ActivePeerDomain =
CheckSum = 3645744851
CommandPath = /usr/local/bin/root_command
ControlFlags = 1
Description =
FilterArg =
FilterScript =
Lock = 0
NodeNameList = {ms_node.network}
RunCmdName = rootcmd

[user01@ms_node]# runlpcmd -N rootcmd
You just ran a command requiring root authority on ms_node.network
RC = 0
```

The user *user01* is able to both list and execute the resource. The Resource ACL contains an entry for `host:user01@LOCALHOST` that grants the needed permissions, **r** and **x**.

Next, the user *user02* tries it.

```
[user02@ms_node]# lslpcmd rootcmdName = rootcmd
ActivePeerDomain =
CheckSum = 3645744851
CommandPath = /usr/local/bin/root_command
ControlFlags = 1
Description =
FilterArg =
FilterScript =
Lock = 0
NodeNameList = {ms_node.network}
RunCmdName = rootcmd
```

```
[user02@ms_node]# runlpcmd -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user02@0x3ea9ab8f7d18ea6e requires 'x' permission for the re
source '0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0' of class I
BM.LPCommands on node ms_node.network.
```

The user *user02* is able to list the LP resource, but not to execute it. The Resource ACL does not contain an entry specific to *user02*. However, it does contain a matching wildcarded HBA ACL entry, `host:LOCALHOST`. This ACL entry grants *user02* **r** permission, but not **x** permission.

Finally, the user *user03* tries it.

```
[user03@ms_node]# lslpcmd rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user03@0x3ea9ab8f7d18ea6e requires 'q' permission for the re
source '0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0' of class I
BM.LPCommands on node ms_node.network.

[user03@ms_node]# runlpcmd -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user03@0x3ea9ab8f7d18ea6e requires 'x' permission for the re
source '0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0' of class I
BM.LPCommands on node ms_node.network.
```

The user *user03* is not able to list or execute the LP resource. The Resource ACL contains an entry for `host:user03@LOCALHOST`. That entry denies all permission to the resource.

Understanding implicit permissions for the mapped root user

If permission to list and change an LP ACL were solely controlled by ACL entries, it would be possible to set an ACL such that no user could modify it again. That is not desirable.

Some user must always have permission to manipulate ACLs. The policy of the RMC subsystem is that the mapped root user will always be allowed to list and change LP ACLs. This policy is realized by implicitly granting the mapped root user **q** (query), **l** (list), and **a** (administration) permissions for class and resource operations.

The identity mappings shipped by RSCT map the HBA root network identity of the local node to the root user, and map the HBA root network identities of any node in an active peer domain to the root user. These mappings appear in the **ctsec_map.global** file in **/usr/sbin/rsct/cfg**:

```
unix:root@<iw>=root
unix:root@<cluster>=root
```

Example: The following scenario shows how the mapped root user can restore ACLs that are set to deny all access.

1. The following command demonstrates that *user01* has permission to run a command through LPRM:

   ```
   [user01@ms_node]# runlpcmd -N rootcmd
   You just ran a command requiring root authority on ms_node.network
   RC = 0
   ```

2. Now the Class ACL and the *rootcmd* Resource ACL are set to deny all access.

   ```
   [root@ms_node]# chlpclacl -x
   [root@ms_node]# chlpracl -x rootcmd

   [root@ms_node]# lslpclaclClass ACLs for LPRM
   Identity          Permissions NodeName
   No access defined             ms_node.network

   [root@ms_node]# lslpracl rootcmd
   Resource ACLs for LPRM
   Name    Identity          Permissions NodeName
   rootcmd No access defined             ms_node.network
   ```

3. The user *user01* can no longer run the command through LPRM. When *user01* tries to do so, access is denied by the Class ACL.

```
[user01@ms_node]# runlpcmd -N rootcmd
2610-441 Permission is denied to access the resource class specified in this
 command.
Network Identity user01@0x3ea9ab8f7d18ea6e requires 'l' permission for the r
esource
class IBM.LPCommands on node ms_node.network.

[user01@ms_node]# lslpclacl
2610-441 Permission is denied to access the resource class specified in this
command.
Network Identity user01@0x3ea9ab8f7d18ea6e requires 'q' permission for the r
esource
class IBM.LPCommands on node ms_node.network.
Class ACLs for LPRM
```

4. The mapped root user is able to restore the Class ACL.

```
[root@ms_node]# chlpclacl root@LOCALHOST rwa LOCALHOST r
```

5. The user *user01* can still not run the command through LPRM. When *user01* tries to do so again, access is denied by the Resource ACL.

```
[user01@ms_node]# runlpcmd -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user01@0x3ea9ab8f7d18ea6e requires 'x' permission for the re
source '0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0' of class I
BM.LPCommands on node ms_node.network.

[user01@ms_node]# lslpracl rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user01@0x3ea9ab8f7d18ea6e requires 'q' permission for the re
source '0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0' of class I
BM.LPCommands on node ms_node.network.
```

6. The mapped root user is able to restore the Resource ACL.

```
[root@ms_node]# chlpracl rootcmd root@LOCALHOST rwa \
user01@LOCALHOST rx LOCALHOST r
```

7. Now, the user *user01* can run the command through LPRM.

```
[user01@ms_node]# runlpcmd -N rootcmd
You just ran a command requiring root authority on ms_node.network
RC = 0
```

Determining the target nodes for an LPRM command

You can run LPRM commands on a single machine, on all the nodes of a peer domain, or on all the nodes of a management domain. The LPRM commands enable you to refine this capability even further, allowing you to specify a subset of nodes in the peer domain or management domain.

Two environment variables that, together with various command flags, determine the nodes that will be affected by the LPRM commands you enter:

CT_CONTACT
: Determines the system that is used for the session with the RMC daemon. When the CT_CONTACT environment variable is set to a host name or IP address, the command contacts the RMC daemon on the specified host. If the environment variable is not set, the command contacts the RMC daemon on the local system where the command is being run. The target of the RMC daemon session and the management scope determine the LP resources that are processed.

CT_MANAGEMENT_SCOPE
: Determines the management scope that is used for the session with the RMC daemon to process the LP resources. The management scope determines the set of possible target nodes where the resources can be processed. The valid values are:

  0 Specifies *local* scope.

1 Specifies *local* scope.
2 Specifies *peer domain* scope.
3 Specifies *management domain* scope.

If this environment variable is not set, *local* scope is used.

Monitoring LP resources and operations

The LP resource manager provides two commands for monitoring LP resources and operations on one node or across many nodes.

Those commands are as follows:

lslpcmd
This command lists the root commands or scripts that are defined as LP resources. Depending on the parameters and flags that you specify, the list contains either the names of LP resources or the names plus attributes of LP resources. To use this LPRM command, you must have read (**r**) permission to the IBM.LPCommands resource class, and read (**r**) permission for the individual IBM.LPCommands resources.

lphistory
This command lists the history of LP commands that were run by the LP resource manager. Through the *NumCommands* parameter, you may specify the number of commands that you want returned in the list. The command history is maintained as records in the RSCT audit log. Depending on the parameter and flags that you specify, the list contains either the command string (path name plus arguments) for the latest number of records, or the specified fields for a selected set of records from one or more nodes. You can also use the **lphistory** command to clear the history of LP commands. To use this LPRM command, you must have write permission to the IBM.LPCommands resource class.

In addition, you may use the dynamic attributes of the IBM.LPCommands resource class to create your own conditions for monitoring. For more information, see "Advanced resource monitoring" on page 152.

The LP resource manager also uses the Audit log resource manager to log detailed usage information about LP resources. For more information about the Audit log resource manager, see the following topics:

- "Resource managers provided with RSCT" on page 110
- "Using the audit log to track monitoring activity" on page 145

Defining LP resources and authorized users

Use the **mklpcmd** command to create an LP resource and, optionally, to set permissions for an LP resource.

- Table 66 lists the permissions you must have specified in the Class ACL and Resource Initial ACL on all nodes where the LP resource will be created. If the Resource Initial ACL directs use of the Resource Shared ACL, then the Resource Shared ACL should have the specified permissions.

Table 66. Required permissions for creating LP resources

Action	Required Class ACL permission	Required Resource Initial ACL permission
Create LP command without specifying permissions	**w**	not applicable
Create LP command and specify permissions	**rw**	**ra**

To determine if you have the correct permissions, check the settings in the Class ACL (as described in "Displaying the Class ACL" on page 228) and the Resource Initial ACL (as described in "Displaying the Resource Initial ACL" on page 230).

- If you are going to specify the optional set of permissions for the resource you are creating, understand the concepts described in "Overview of the LP resource manager's access control lists" on page 215.

- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables on the **mklpcmd** command. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

Perform the following steps to define an LP resource and its authorized users:

1. Determine which users require access to this LP resource.

   A Resource ACL for the new resource will be created when you run the **mklpcmd** command. Decide whether or not you will need to specify an optional set of permissions on the **mklpcmd** command, or whether the default entries in the Resource Initial ACL are sufficient. The entries of the Resource ACL are copied from the Resource Initial ACL and then modified by the ACL entries optionally specified with the **mklpcmd** command. You can later modify the Resource ACL as described in "Modifying a Resource ACL" on page 229.
2. Determine the location where the root command or script will reside. You will need the fully qualified path of the command or script and, optionally, the nodes on which it will be available.
3. Determine whether you want the LP resource manager to validate the command or script whenever a user issues an LPRM command for the resource you are defining. This decision determines whether you use the default or specify a value for the ControlFlags attribute for the LP resource.
4. Issue the **mklpcmd** command, supplying appropriate values for required parameters and flags.

   Example: To define a new LP resource, named *LP1*, pointing to the command */tmp/user1/lpcmd1* on a local node, you would enter:

   ```
   mklpcmd LP1 /tmp/user1/lpcmd1
   ```

   If you want to define permissions required for a user to run the root command or script that is defined as the LP resource, include user identifiers and permission specifiers following the command path. A user will require execute (**x**) permission to run the root command or script through the **runlpcmd** command.

   Example: To create an LP resource called *LP5* that points to */usr/bin/mkrsrc* and grants users *user1@LOCALHOST* and *user2@LOCALHOST* read and execute permission, enter:

   ```
   mklpcmd LP5 /usr/bin/mkrsrc user1@LOCALHOST rx user2@LOCALHOST rx
   ```

You know you are done when the LP resource manager returns an exit value or message from processing the command or script.

Running an LP resource

Use the **runlpcmd** command to run a root command or script that is defined as an LP resource.

- Table 67 lists the permissions you must have specified in the Class ACL and Resource ACL. If the Resource ACL directs use of the Resource Shared ACL, then the Resource Shared ACL should have the specified permissions.

Table 67. Required permissions for running an LP resource

Required Class ACL permission	Required Resource ACL permission
r	**x**

  To determine if you have the correct permissions, check the settings in the Class ACL (as described in "Displaying the Class ACL" on page 228) and the Resource ACL (as described in "Displaying a Resource ACL" on page 229).
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables on the **runlpcmd** command. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

Perform the following step to run an LP resource.

Issue the **runlpcmd** command, supplying appropriate values for required parameters and flags.
Example: To run the LP resource named *LP1*, which has required input flags and parameters **-a -p User Group**, you would enter:

```
runlpcmd LP1 "-a -p User Group"
```

You know you are done when the LP resource manager returns an exit value or message from processing the command or script.

For complete syntax information on the **runlpcmd** command, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying an LP resource

Use the **chlpcmd** command to modify an LP resource.

- Table 68 lists the permissions you must have specified in the Class ACL and Resource ACL on all nodes where the LP resource will be modified. If the Resource ACL directs use of the Resource Shared ACL, then the Resource Shared ACL should have the specified permissions.

Table 68. Required permissions for modifying an LP resource

Required Class ACL permission	Required Resource ACL permission
r	**w**

  To determine if you have the correct permissions, check the settings in the class ACL (as described in "Displaying the Class ACL" on page 228) and the Resource ACL (as described in "Displaying a Resource ACL" on page 229).
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables on the **chlpcmd** command. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To modify an LP resource, issue the **chlpcmd** command, supplying appropriate values for required parameters and flags.

Example: To change the Lock attribute of an LP resource named *LP1*, you would enter:

```
chlpcmd LP1 Lock=0
```

You know you are done when the LP resource manager returns an exit value or message from processing the command or script.

Removing LP resources

Use the **rmlpcmd** command to remove an LP resource.

- You need to have read (**r**) and write (**w**) permission specified in the Class ACL on all nodes where the LP resource will be removed. To determine if you have the correct permissions, see the setting in the Class ACL as described in "Displaying the Class ACL" on page 228).
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables on the **rmlpcmd** command. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

Perform the following steps to remove an LP resource.

1. (*Optional*) Use the **lslpcmd** command to display the attribute values for this LP resource. If the resource is locked, you must change the Lock attribute value to 0 before attempting to remove the resource.
2. Issue the **rmlpcmd** command, supplying appropriate values for required parameters and flags.

   Example: To remove the LP resource named LP1, you would enter:

   ```
   rmlpcmd LP1
   ```

Result: The LP resource manager returns an exit value or message from processing the command or script.

For complete syntax information on the **lslpcmd** and **rmlpcmd** commands, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Displaying the Class ACL

Use the **lslpclacl** command to display the access controls for the IBM.LPCommands resource class.

- To display the Class ACL, you need to have read (**r**) permission specified in the Class ACL, or else you need to be the mapped root identity.
- Understand the purpose of the Class ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To display the Class ACL, issue the **lslpclacl** command.

Example:

```
lslpclacl
```

Result: The following output is displayed:

```
Class ACLs for LPRM
Identity            Permissions NodeName
host:root@LOCALHOST rwa         nodeA
host:LOCALHOST      r           nodeA
```

For complete syntax information on the **lslpclacl** command (including additional flags you can specify), see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying the Class ACL

Use the **chlpclacl** command to change the access controls for the IBM.LPCommands resource class.

- To modify the Class ACL, you need to have read (**r**) and administration (**a**) permission specified in the Class ACL. Alternatively, you need to be the mapped root identity. To determine if you have the correct permissions, check the settings in the Class ACL as described in "Displaying the Class ACL").
- Understand the purpose of the Class ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To modify the Class ACL, issue the **chlplacl** command, supplying appropriate values for parameters and flags.

Example: To give user *lpadmin* read and write permission to the IBM.LPCommands class on the local node so that he or she can list and create LP resources on the node, run the following command:

```
chlpclacl lpadmin@LOCALHOST rw
```

For complete syntax information on the **chlplacl** command (including additional flags you can specify), see its online man page.

Displaying a Resource ACL

Use the **lslpracl** command to display the access controls for a resource of the IBM.LPCommands resource class.

- To display a Resource ACL, obtain the following permissions specified in the Class ACL and Resource ACL. Alternatively, if you are the mapped root identity, you can display a Resource ACL . If the Resource ACL directs use of the Resource Shared ACL, then the Resource Shared ACL will require the specified permissions.

Required Class ACL permission	Required Resource ACL permission
r	r

- Understand the purpose of a Resource ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To display the Resource ACL, issue the **lslpracl** command with the name of the resource as in the following example.

Example:

```
lslpracl LPCommand1
```

Result: The following output is displayed:

```
Resource ACLs for LPRM
Name       Identity            Permissions NodeName
LPCommand1 host:root@LOCALHOST rwa         nodeA
LPCommand1 host:joe@LOCALHOST  rx          nodeA
LPCommand1 host:LOCALHOST      r           nodeA
```

If the *Identity* column of the output reads Uses Resource Shared ACL, this means that the Resource Shared ACL is being used to control access to this resource. To display the access controls in the Resource Shared ACL, see "Displaying a Resource Shared ACL" on page 231 or issue the **lslpracl** command with its **-L** flag as in the following example.

Example:

```
lslpracl -L LPCommand1
```

For complete syntax information on the **lslpracl** command (including additional flags you can specify), see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying a Resource ACL

Use the **chlpracl** command to change the access controls for a resource of the IBM.LPCommands resource class.

- To modify a Resource ACL, you need the following permissions specified in the Class ACL and Resource ACL. Alternatively, you can modify a Resource ACL if you are the mapped root identity. If the Resource ACL directs use of the Resource Shared ACL, then the Resource Shared ACL should have the specified permissions.

Required Class ACL permission	Required Resource ACL permission
r	**ra**

- Understand the purpose of the Resource ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To modify the Resource ACL, issue the **chlpracl** command, supplying appropriate values for parameters and flags.

Example: To allow user *joe* on the local node to list and run the LP resource *LPcommand1* on the local node, run the following command.

```
chlpracl LPcommand1 joe@LOCALHOST rx
```

For complete syntax information on the **chlplacl** command (including additional flags you can specify), see its online man page.

Displaying the Resource Initial ACL

Use the **lslpriacl** command to display the access controls for the Resource Initial ACL.

- To display the Resource Initial ACL, you need to have read (**r**) permission specified in the Class ACL. Alternatively, you need to be the mapped root identity.
- Understand the purpose of the Resource Initial ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To display the Resource Initial ACL, issue the **lslpriacl** command as in the following example.

Example:

```
lslpriacl
```

Result: The following output is displayed:

```
Resource Initial ACLs for LPRM
Identity            Permissions NodeName
host:root@LOCALHOST rwa         nodeA
host:LOCALHOST      r           nodeA
```

For complete syntax information on the **lslpriacl** command (including additional flags you can specify), see its online man page. For detailed syntax information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Modifying the Resource Initial ACL

Use the **chlpriacl** command to change the access controls for the Resource Initial ACL.

- To modify the Resource Initial ACL, you need to have read (**r**) and administration (**a**) permission specified in the Class ACL. Alternatively, you need to be the mapped root identity.
- Understand the purpose of the Resource Initial ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To modify the Resource Initial ACL, issue the **chlpriacl** command, supplying appropriate values for parameters and flags.

Example: To give user *joe* on the local node read (**r**) and execute (**x**) permission in the Resource Initial ACL, run the following command:

```
chlpriacl joe@LOCALHOST rx
```

For complete syntax information on the **chlpriacl** command (including additional flags you can specify), see its online man page.

Displaying a Resource Shared ACL

Use the **lslprsacl** command to display the access controls for Resource Shared ACL.

- To display the Resource Shared ACL, you need to have read (**r**) permission specified in the Class ACL. Alternatively, you need to be the mapped root identity.
- Understand the purpose of the Resource Shared ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To display the Resource Shared ACL, issue the **lslprsacl** command as in the following example.

Example:

```
lslprsacl
```

Result: The following output is displayed:

```
Resource Shared ACLs for LPRM
Identity            Permissions NodeName
host:root@LOCALHOST rwa         nodeA
host:LOCALHOST      r           nodeA
```

For complete syntax information on the **lslprsacl** command (including additional flags you can specify), see its online man page.

Modifying a Resource Shared ACL

Use the **chlprsacl** command to change the access controls for the Resource Shared ACL.

- To modify the Resource Shared ACL, you need to have read (**r**) and administration (**a**) permission specified in the Class ACL, or else you need to be the mapped root identity.
- Understand the purpose of the Class ACL as described in "Overview of the LP resource manager's access control lists" on page 215.
- Determine what values to set for the CT_CONTACT and CT_MANAGEMENT_SCOPE environment variables. To do so, use the information in "Determining the target nodes for an LPRM command" on page 224.

To modify the Class ACL, issue the **chlprsacl** command, supplying appropriate values for parameters and flags.

Example: To give user *joe* on the local node read (**r**) and execute (**x**) permission in the Resource Shared ACL, run the following command:

```
chlpriacl joe@LOCALHOST rx
```

For complete syntax information on the **chlprsacl** command (including additional flags you can specify), see its online man page.

Specifying host based authentication ACL entries

LP ACL entries may specify network identifiers provided by the host based authentication (HBA) security mechanism.

These are described in "Overview of the LP resource manager's access control lists" on page 215. HBA network identifiers specify a host (or node) in a variety of forms. The host can be specified using a fully qualified host name, an IP address, or an RSCT node ID. These forms are not interchangeable.

When a client connects to the RMC subsystem, the client is authenticated and a network identifier representing the client is generated. When the HBA security mechanism is used to authenticate the client, the host component of the network identifier will be one of the forms listed above. The form that is used depends on how the client connected to the RMC subsystem.

From the point of view of the command line interface, the type of connection made to the RMC subsystem depends on the values of CT_CONTACT and CT_IP_AUTHENT in the client's environment.

1. If the CT_CONTACT environment variable is not set, the client command connects to the local RMC daemon, and the HBA network identifier for the client specifies the RSCT node ID of the node.
2. If the CT_CONTACT environment variable is set, the client command connects remotely to the RMC daemon running on the node specified by CT_CONTACT.
   a. If the CT_IP_AUTHENT environment variable is not set, the HBA network identifier for the client specifies the fully qualified host name of the node running the client command.
   b. If the CT_IP_AUTHENT environment variable is set, the HBA network identifier for the client specifies the IP address of the node running the client command.

When an authenticated RMC client attempts to execute an RMC class or resource operation, an ACL is checked to determine if the client has the necessary authority to perform the operation. If the HBA security mechanism had been used to authenticate the client, the only HBA ACL entries that are considered during an authorization check are those utilizing the same form for the host component of the network identifier.

As can be seen from the prior discussion, when setting up LP ACLs with HBA ACL entries, it is important to understand how you expect clients to connect to the RMC subsystem. How clients are expected to connect to the RMC subsystem affects how hosts should be specified in HBA ACL entries.

Examples of Host Based Authentication ACL entries

The following examples involve two nodes.

Node **ms_node** is a management server and **nodeA** is one of its managed nodes.

Two users are used in the examples, **root** and **user01**. The shell prompts show the user running the commands and the node on which the commands are run.

For the purposes of the examples, a simple shell script is used. The shell script displays a message that includes the host name of the node on which the script is run. This script is placed on both the nodes involved in these examples. The file permissions of the script are set such that only the root user can execute the script without the use of LPRM. The following commands show this arrangement on node *ms_node*.

```
[root@ms_node]# ls -l /usr/local/bin/root_command
-rwx------    1 root   root   109 Mar  3 18:22 /usr/local/bin/root_command

[root@ms_node]# cat /usr/local/bin/root_command#!/bin/ksh
thishost=$(/bin/hostname)
/bin/echo "You just ran a command requiring root authority on $thishost"
```

From *ms_node*, the root user defines the *root_command* script as an LP resource on both the nodes.

```
[root@ms_node]# mklpcmd rootcmd /usr/local/bin/root_command
[root@ms_node]# mklpcmd -n nodeA rootcmd /usr/local/bin/root_command
```

Example of local connections in local scope:

The following commands show *user01* on *ms_node* attempting to run the *root_command* script locally through LPRM. The attempt fails.

```
[user01@ms_node]# echo $CT_CONTACT

[user01@ms_node]# echo $CT_MANAGEMENT_SCOPE

[user01@ms_node]# runlpcmd -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user01@0x3ea9ab8f7d18ea6e requires 'x' permission for the re
source '0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0' of class I
BM.LPCommands on node ms_node.network.
```

The error message indicates which specific network identity requires which specific permission for which specific resource on which specific node. Notice that the network identity specifies the host using the RSCT node ID. The following command confirms that this is the RSCT node ID of the local host.

```
[user01@ms_node]# /usr/sbin/rsct/bin/lsnodeid
3ea9ab8f7d18ea6e
```

The following commands show the root user adding the required permission for *user01* to run *root_command* locally through LPRM.

```
[root@ms_node]# lsrsrc -s "Name='rootcmd'" IBM.LPCommands Name ResourceHandle
Resource Persistent Attributes for IBM.LPCommands
resource 1:
Name           = "rootcmd"
ResourceHandle = "0x6040 0xffff 0x3ea9ab8f 0x7d18ea6e 0x0f6b6d2c 0x4dae62b0"

[root@ms_node]# lslpracl rootcmdResource ACLs for LPRM
Name    Identity            Permissions NodeName
rootcmd host:root@LOCALHOST rwa         ms_node.network
rootcmd host:LOCALHOST      r           ms_node.network

[root@ms_node]# chlpracl rootcmd user01@LOCALHOST rx
[root@ms_node]# lslpracl rootcmdResource ACLs for LPRM
Name    Identity              Permissions NodeName
rootcmd host:root@LOCALHOST   rwa         ms_node.network
rootcmd host:user01@LOCALHOST rx          ms_node.network
rootcmd host:LOCALHOST        r           ms_node.network
```

The following command shows *user01* can now run *root_command* locally through LPRM.

```
[user01@ms_node]# runlpcmd -N rootcmd
You just ran a command requiring root authority on ms_node.network
RC = 0
```

Example of local connections in management domain scope:

When operating in management domain scope on a management server, connections to the RMC subsystem are typically local. The RMC subsystem forwards requests to the managed nodes as needed.

In this example, **user01** on **ms_node** is attempting to run **root_command** on **nodeA** through LPRM. The connection to the RMC subsystem is local on **ms_node**.

First, the attempt fails because permissions are not set up on **nodeA** to allow this to happen.

```
[user01@ms_node]# runlpcmd -n nodeA -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user01@0x3ea9ab8f7d18ea6e requires 'x' permission for the
resource '0x6040 0xffff 0xbb5dca56 0xfe5aa5e8 0x0f6b6f40 0xbf598b90' of class
IBM.LPCommands on node nodeA.network.
```

Next, the root user on **ms_node** can remotely change the appropriate Resource ACL on **nodeA**. Notice that in this example the **NODEID** keyword is used instead of the **LOCALHOST** keyword. **NODEID** is used because the ACL entry being added must reference the node on which the ACL editing command is being run (**ms_node**), not the node on which the ACL is stored (**nodeA**).

```
[root@ms_node]# chlpracl -n nodeA rootcmd user01@NODEID rx

[root@ms_node]# lslpracl -n nodeA rootcmdResource ACLs for LPRM
Name    Identity                         Permissions NodeName
rootcmd host:root@0x3ea9ab8f7d18ea6e   rwax        nodeA.network
rootcmd host:root@192.168.46.107       rwax        nodeA.network
rootcmd host:root@ms_node.network      rwax        nodeA.network
rootcmd host:root@LOCALHOST            rw          nodeA.network
rootcmd host:user01@0x3ea9ab8f7d18ea6e rx          nodeA.network
rootcmd host:LOCALHOST                 r           nodeA.network
rootcmd host:ms_node.network           r           nodeA.network
rootcmd host:192.168.46.107            r           nodeA.network
rootcmd host:0x3ea9ab8f7d18ea6e        r           nodeA.network
```

Finally, **user01** on **ms_node** can execute **root_command** on **nodeA** through LPRM.

```
[user01@ms_node]# runlpcmd -n nodeA -N rootcmd
You just ran a command requiring root authority on nodeA.network
RC = 0
```

Example of remote connections:

In this example, *user01* attempts to run the *root_command* script remotely using a remote RMC connection.

```
[user01@ms_node]# CT_CONTACT=nodeA runlpcmd -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user01@ms_node.network requires 'x' permission for the resou
rce '0x6040 0xffff 0xbb5dca56 0xfe5aa5e8 0x0f6b6f40 0xbf598b90' of class IBM.
LPCommands on node nodeA.network.
```

The ACL entry that had been set up on *nodeA* in the prior example, which identified the *ms_node* host by its RSCT node ID, is not allowing *user01* to execute *root_command* on *nodeA* in this case. As can be seen in the error message, in this case an ACL entry is needed that identifies the host by its host name.

The root user changes the *rootcmd* Resource ACL on *nodeA*:

```
[root@ms_node]# chlpracl -n nodeA rootcmd user01@ms_node.network rx

[root@ms_node]# lslpracl -n nodeA rootcmdResource ACLs for LPRM
Name    Identity                         Permissions NodeName
rootcmd host:root@0x3ea9ab8f7d18ea6e   rwax        nodeA.network
rootcmd host:root@192.168.46.107       rwax        nodeA.network
rootcmd host:root@ms_node.network      rwax        nodeA.network
rootcmd host:root@LOCALHOST            rw          nodeA.network
rootcmd host:user01@0x3ea9ab8f7d18ea6e rx          nodeA.network
rootcmd host:user01@ms_node.network    rx          nodeA.network
rootcmd host:LOCALHOST                 r           nodeA.network
rootcmd host:ms_node.network           r           nodeA.network
rootcmd host:192.168.46.107            r           nodeA.network
rootcmd host:0x3ea9ab8f7d18ea6e        r           nodeA.network
```

Now, *user01* on *ms_node* can run the *root_command* script remotely using a remote RMC connection.

```
[user01@ms_node]# CT_CONTACT=nodeA runlpcmd -N rootcmd
You just ran a command requiring root authority on nodeA.network
RC = 0
```

Example of remote connections (using IP addresses):

In this example, as in the previous example, *user01* attempts to run the *root_command* script remotely using a remote RMC connection.

However, this time the client specifies that authentication is to be done using IP addresses, not host names.

```
[user01@ms_node]# CT_IP_AUTHENT=1 CT_CONTACT=192.168.46.108 runlpcmd -N rootcmd
2610-440 Permission is denied to access a resource specified in this command.
Network Identity user01@192.168.46.107 requires 'x' permission for the resource
 '0x6040 0xffff 0xbb5dca56 0xfe5aa5e8 0x0f6b6f40 0xbf598b90' of class IBM.LPCom
mands on node nodeA.network.
```

The ACL entry that had been set up on *nodeA* in the prior example, which identified the *ms_node* host by its host name, is not allowing *user01* to execute *root_command* on *nodeA* in this case. As can be seen in the error message, in this case an ACL entry is needed that identifies the host by its IP address.

The root user changes the *rootcmd* Resource ACL on *nodeA*:

```
[root@ms_node]# chlpracl -n nodeA rootcmd user01@192.168.46.107 rx

[root@ms_node]# lslpracl -n nodeA rootcmdResource ACLs for LPRM
Name    Identity                        Permissions NodeName
rootcmd host:root@0x3ea9ab8f7d18ea6e   rwax        nodeA.network
rootcmd host:root@192.168.46.107       rwax        nodeA.network
rootcmd host:root@ms_node.network      rwax        nodeA.network
rootcmd host:root@LOCALHOST            rw          nodeA.network
rootcmd host:user01@0x3ea9ab8f7d18ea6e rx          nodeA.network
rootcmd host:user01@192.168.46.107     rx          nodeA.network
rootcmd host:user01@ms_node.network    rx          nodeA.network
rootcmd host:LOCALHOST                 r           nodeA.network
rootcmd host:ms_node.network           r           nodeA.network
rootcmd host:192.168.46.107            r           nodeA.network
rootcmd host:0x3ea9ab8f7d18ea6e        r           nodeA.network
```

Now, *user01* on *ms_node* can run the *root_command* script remotely using a remote RMC connection that uses IP address authentication.

```
[user01@ms_node]# CT_IP_AUTHENT=1 CT_CONTACT=192.168.46.108 runlpcmd -N rootcmd
You just ran a command requiring root authority on nodeA.network
RC = 0
```

Restricted execution based on command arguments

Once it is determined that a user has the authority to execute a command through the LPRM **runlpcmd** command, it may be necessary to determine if the user is permitted to run the command using the arguments he or she has specified.

It is not possible for the RMC subsystem or LPRM to make that determination directly, since neither RMC nor LPRM is aware of the meaning of the arguments of the command specified by the LP resource.

The FilterScript and FilterArg attributes of an IBM.LPCommands resource allow for restricted execution based on command arguments.

- The FilterScript resource attribute is a character string that specifies the full path name of a filter script. A filter script accepts two inputs — a string representing permitted arguments, and a string representing the command arguments specified by the user with the **runlpcmd** command. The job of the filter script is to determine if the specified arguments are acceptable based on the permitted arguments.

- The FilterArg resource attribute is a character string that specifies the permitted arguments. This does not necessarily specify all the arguments allowed by the resource's command. Rather, it specifies the arguments that are permitted to be specified for the command when it is executed through LPRM using that particular resource.

When a user attempts to run a command using **runlpcmd**, the filter script is run first. If the filter script indicates the user-specified arguments are not permitted, LPRM will not run the command. If the filter script indicates the user-specified arguments are permitted, LPRM will run the command.

The FilterScript resource attribute value may be an empty string, indicating there is no filter script to run for the command.

Example: Suppose the **IBM.LPCommands** class resources shown in Figure 3 on page 237 exist on a management server, *ms_node*:

```
[root@ms_node]# lslpcmd rpower_bld Name = rpower_bld
 ActivePeerDomain =
 CheckSum = 2480571332
 CommandPath = /opt/csm/bin/rpower
 ControlFlags = 1
 Description =
 FilterArg = -n node1,node2,node3
 FilterScript = /opt/csm/samples/security/CSMCmdFilter
 Lock = 0
 NodeNameList = {ms_node.network}
 RunCmdName = rpower

[root@ms_node]# lslpracl rpower_bldResource ACLs for LPRM
Name       Identity                  Permissions NodeName
rpower_bld host:bld_admin1@LOCALHOST rx          ms_node.network
rpower_bld host:bld_admin2@LOCALHOST rx          ms_node.network
rpower_bld host:root@LOCALHOST       rwa         ms_node.network
rpower_bld host:LOCALHOST            r           ms_node.network

[root@ms_node]# lslpcmd rpower_prod Name = rpower_prod
 ActivePeerDomain =
 CheckSum = 2480571332
 CommandPath = /opt/csm/bin/rpower
 ControlFlags = 1
 Description =
 FilterArg = -n node4,node5
 FilterScript = /opt/csm/samples/security/CSMCmdFilter
 Lock = 0
 NodeNameList = {ms_node.network}
 RunCmdName = rpower

[root@ms_node]# lslpracl rpower_prodResource ACLs for LPRM
Name        Identity                   Permissions NodeName
rpower_prod host:prod_admin1@LOCALHOST rx          ms_node.network
rpower_prod host:prod_admin2@LOCALHOST rx          ms_node.network
rpower_prod host:root@LOCALHOST        rwa         ms_node.network
rpower_prod host:LOCALHOST             r           ms_node.network

[root@ms_node]# lslpcmd rpower_any Name = rpower_any
 ActivePeerDomain =
 CheckSum = 592515412
 CommandPath = /opt/csm/bin/rpower
 ControlFlags = 1
 Description =
 FilterArg =
 FilterScript =
 Lock = 0
 NodeNameList = {ms_node.network}
 RunCmdName = rpower

[root@ms_node]# lslpracl rpower_anyResource ACLs for LPRM
Name       Identity                  Permissions NodeName
rpower_any host:root@LOCALHOST       rwa         ms_node.network
rpower_any host:super_admin@LOCALHOST rx         ms_node.network
rpower_any host:LOCALHOST            r           ms_node.network
```

Figure 3. Example of IBM.LPCommands resources on management server ms_node

All these resources are defined to allow for the execution of the **rpower** command through LPRM (refer to the value of the CommandPath attribute).

- The Resource ACL of the resource whose Name attribute has the value *rpower_bld* allows users *bld_admin1* and *bld_admin2* on the management server to execute **rpower**.
- The Resource ACL of the resource whose Name attribute has the value *rpower_prod* allows users *prod_admin1* and *prod_admin2* on the management server to execute **rpower**.

- The Resource ACL of the resource whose Name attribute has the value *rpower_any* allows the *super_admin* user on the management server to execute **rpower**.

Note that the *rpower_any* resource has an empty string value for the FilterScript attribute. This means that when a user executes the **rpower** command through LPRM using this resource there are no restrictions to the arguments that may be specified. This is in contrast to the *rpower_bld* and *rpower_prod* resources. Both these resources specify the path name to some filter script.

For the purposes of this example, assume the *CSMCmdFilter* filter script expects the FilterArg value to be a specification of nodes that are permitted to be targeted by the **rpower** command. The value of the FilterArg attribute for the *rpower_bld* resource then indicates that *node1*, *node2*, and/or *node3* may be targeted. The value of the FilterArg attribute for the *rpower_prod* resource indicates that *node4* and/or *node5* may be targeted.

Now, we will illustrate the combined effect of these resources, and specifically of their Resource ACL, FilterScript, and FilterArg values.

- The *super_admin* user can execute **rpower** through LPRM, targeting any managed node, using the *rpower_any* resource. He can do this because the Resource ACL of the resource gives him permission to execute the **rpower** command, and the absence of a filter script for the resource means there are no restrictions on what nodes he can target with the command.
- The *bld_admin1* user can execute **rpower** through LPRM, targeting *node1*, *node2*, and/or *node3*, using the *rpower_bld* resource. He can execute the **rpower** command because the Resource ACL of the resource allows it. He is limited to targeting *node1*, *node2*, and/or *node3* because of the values of the FilterScript and FilterArg attributes for the resource.

  The *bld_admin1* user cannot execute **rpower** through LPRM using any other defined resource, because the Resource ACLs of those resources do not give him permission.
- The *prod_admin1* user can execute **rpower** through LPRM, targeting *node4* and/or *node5*, using the *rpower_prod* resource. He can execute the **rpower** command because the Resource ACL of the resource allows it. He is limited to targeting *node4* and/or *node5* because of the values of the FilterScript and FilterArg attributes for the resource.

  The *prod_admin1* user cannot execute *rpower* through LPRM using any other defined resource, because the Resource ACLs of those resources do not give him permission.

Run Command Name for IBM.LPCommands

Because of the filter script capabilities, it is likely that multiple resources will refer to the same command.

These resources will be functionally distinguished from each other by the values of their FilterScript and FilterArg attributes, and their Resource ACL entries. These resources will have unique values for the Name attribute.

While it is possible for the LPRM commands to target resources based on the Name attribute value, it seems less desirable to do so for the **runlpcmd** command.

Example: Consider the example shown in Figure Figure 3 on page 237 in "Restricted execution based on command arguments" on page 235. It seems less than desirable for the administrator who has set up the resources in this example to instruct *prod_admin1* to execute:

```
# runlpcmd -N rpower_prod -n <node_list> off
```

while instructing *bld_admin1* to execute:

```
# runlpcmd -N rpower_bld -n <node_list> off
```

Instructing everyone to execute the following command seems more desirable:

```
# runlpcmd rpower -n <node_list> off
```

The RunCmdName attribute of the individual IBM.LPCommands resources make this possible. Resources that specify the same command, but differ in Resource ACL and FilterScript and FilterArg attribute values, should have the same value for RunCmdName. In the example in "Restricted execution based on command arguments" on page 235, all the LP resources had the RunCmdName of *rpower*.

Now, when the **runlpcmd** command is run without the **-N** flag, specifying the LP resource *rpower*, the resource whose RunCmdName attribute is *rpower* to which the user has execute permission is selected to be run. The **runlpcmd** command will then use that resource, along with that resource's filter script and filter arguments.

Limitations of the Run Command Name

It is necessary to impose a restriction on the use of the RunCmdName attribute to identify which resource is to be used to run a command through LPRM.

If a user has execute permission for multiple resources of the **IBM.LPCommands** class that have the **RunCmdName** value specified with an invocation of **runlpcmd**, **runlpcmd** will not run any command through LPRM. Without this restriction it would not be possible to do the right thing under certain conditions.

Example: Consider the example shown in Figure Figure 3 on page 237 in "Restricted execution based on command arguments" on page 235. If the user *power_admin* is to be given authority to target production and build nodes with the **rpower** command, one might be tempted to issue the following commands:

```
# chlpracl rpower_prod host:power_admin@LOCALHOST rx
# chlpracl rpower_bld  host:power_admin@LOCALHOST rx
```

Once this was done, consider what might happen if the *power_admin* user executed the following command.

```
# runlpcmd rpower -n node3,node4
```

If the **runlpcmd** command tried to use the resource with a **Name** attribute of *rpower_bld*, the filter script for that resource would not allow it, because *node4* is not a permitted argument for that resource. If the **runlpcmd** command tried to use the resource whose **Name** attribute is *rpower_prod*, the filter script for that resource would not allow it, because *node3* is not a permitted argument for that resource. Therefore, the restriction is in place.

Once the **chlpracl** commands shown above are executed, the *power_admin* user cannot run **rpower** using **runlpcmd** specifying a RunCmdName value. Instead, he would have to specify the Name value using the **runlpcmd** command's **-N** flag.

In this case, instead of changing the Resource ACLs of the *rpower_prod* and *rpower_bld* resources, a course of action that would allow the use of the **RunCmdName** value with **runlpcmd** would be to give execute permission to the *power_admin* user for the *rpower_any* resource.

The administrator, or management software (such as CSM), that defines resources in this class should take care to give a user execute permission to only one resource with a specific **RunCmdName** value.

Administering the storage resource manager

The storage resource manager provides monitoring and control for storage resources within an RSCT peer domain.

The storage resource manager provides the interface between RMC and the physical and logical storage entities within the peer domain by mapping these entities to instances of the resource classes it provides. Running as a daemon process on each node in the peer domain, the Storage resource manager collects information about locally-attached physical disks (and related storage entities) and maps these to resource class instances. These separate views of the storage resources from each individual node are then

collected together to provide the Storage resource manager with a global view of the storage resources within an RSCT peer domain. In a shared storage environment (in which multiple nodes within the peer domain have access to the same disk subsystem and therefore access to the same disks contained within the disk subsystem), the storage resource manager's global view of storage resources enables it to identify such shared disks and provide serial access through hardware reserves.

How the storage resource manager gathers and represents storage data

The storage resource manager provides a set of resource classes that enable you to manage and monitor storage resources in an RSCT peer domain.

The storage resource manager provides resource classes to represent a physical disk and related storage entities (such as the volume group to which the disk belongs, logical volumes into which the volume group is divided, and file systems on logical volumes or disk partitions). The storage resource manager runs as a daemon process on each node in the peer domain and, upon peer domain startup, automatically *harvests* (detects and collects) information about locally attached physical disks and their related logical storage entities. The collected information will be mapped to instances of the storage resource manager's resource classes.

The storage resource manager uses the following resource classes to represent the various storage entities:

Disk resource (IBM.Disk)
: This resource class externalizes the attributes of SCSI disks and MD devices on Linux and physical volumes that are members of a volume group.

Volume group resource (IBM.VolumeGroup)
: This resource class externalizes the attributes of volume groups comprised of one or more physical volumes.

Logical volume resource (IBM.LogicalVolume)
: This resource class externalizes the attributes of logical volumes configured in volume groups.

Disk partition resource (IBM.Partition)
: On Linux nodes only, this resource class externalizes the attributes of any configured partitions on a disk device resource of the IBM.Disk class.

File system resource (IBM.AgFileSystem)
: This resource class externalizes the attributes of any file systems on a Linux disk partition resource (IBM.Partition), an entire Linux disk resource (IBM.Disk), or a Linux or AIX logical volume resource (IBM.LogicalVolume). These attributes are a subset of the entries in the IBM.FileSystem class of the File System resource manager.

Because the relationship between storage entities can differ on AIX and Linux nodes, the way in which those entities map to the resource classes also differs.

On Linux nodes, disk resources can contain partition resources which contain file system resources, or a file system can be contained on a disk. These relationships are mapped to IBM.Disk, IBM.Partition, and IBM.AgFileSystem resources, as illustrated in Figure 4 on page 241.

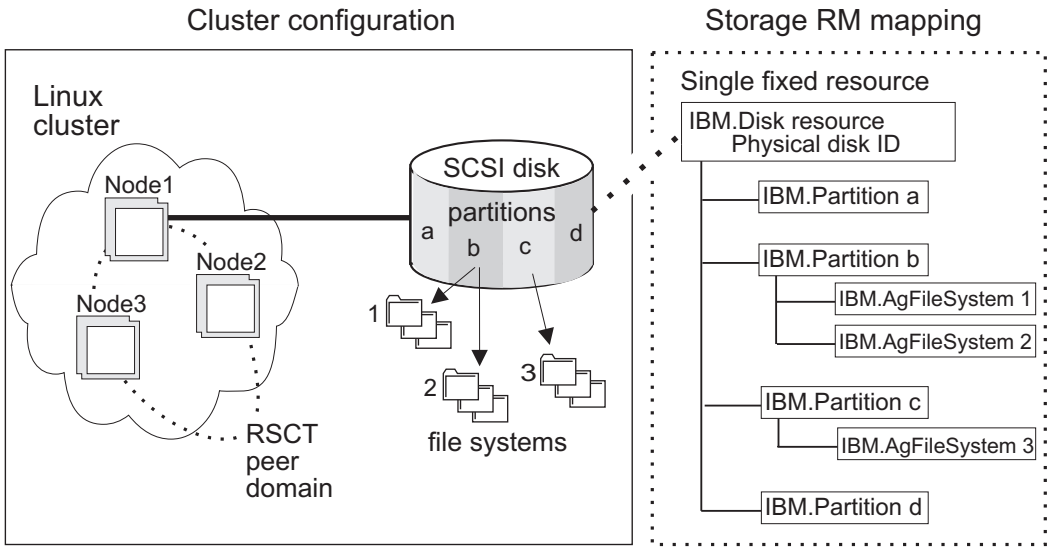

Figure 4. Mapping of storage entities to Storage RM resources using disk partitions on Linux

The relationship between the storage entities shown in Figure 4 are represented using attributes of the IBM.Disk, IBM.Partition, and IBM.AgFileSystem resources, and this information can be obtained using the **lsrsrc** command. See "Discerning the relationship between storage entities on Linux nodes using disk partitions" on page 260 for more information.

On AIX or Linux nodes, volume group resources (each dependent on one or more IBM.Disk resources) contain logical volume resources which contain file system resources. These relationships are mapped to IBM.Disk, IBM.VolumeGroup, IBM.LogicalVolume, and IBM.AgFileSystem resources as illustrated in Figure 5.

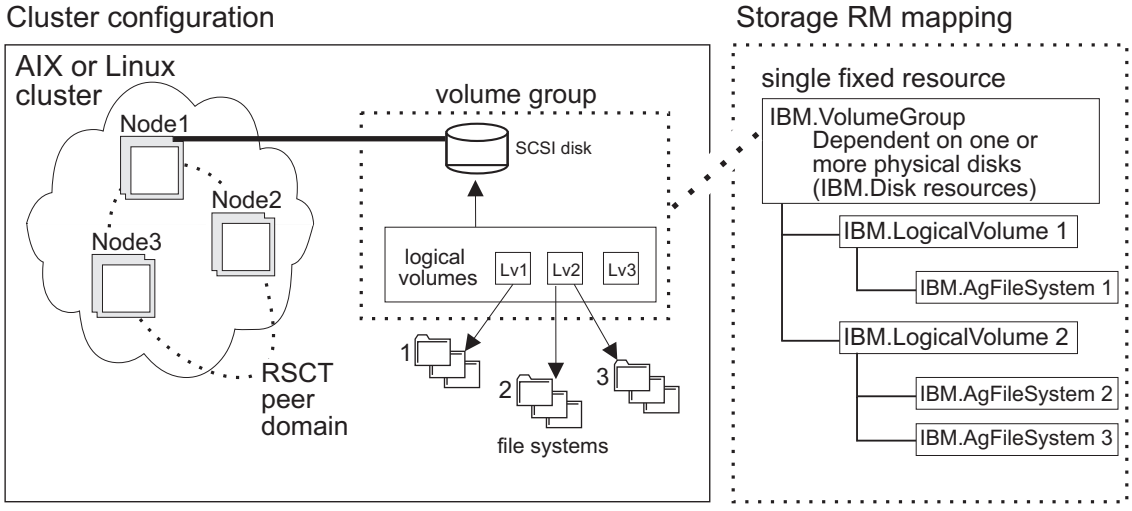

Figure 5. Mapping of storage entities to Storage RM resources using volume groups on AIX or Linux

The relationship between the storage entities shown in Figure 5 are represented using attributes of the IBM.Disk, IBM.VolumeGroup, IBM.LogicalVolume, and IBM.AgFileSystem resources, and this information can be obtained using the **lsrsrc** command. See "Discerning the relationship between storage entities on AIX or Linux nodes using volume groups" on page 262 for more information.

So far, we have described resources that are specific to a single node (called *fixed resources*). Instances of the storage resource manager resource classes can also represent global (or *aggregate*) resources that are shared by more than one node. Once the storage resource manager daemons on each node have harvested their local storage resources and a quorum is achieved within the peer domain, storage configuration information is then exchanged and correlated. This gives the storage resource manager a global view of the entire peer domain's storage resources and enables it to identify physical disks that are attached to multiple nodes in the peer domain. In these situations, there will be one IBM.Disk resource (as well as one of each associated storage resource) for each node through which the disk may be accessed. To manage such shared resources more effectively, the storage resource manager will, by default, create a single *aggregate* resource for each of the shared resources. Then, each fixed resource that represents the same storage entity is called a *constituent resource* of the *aggregate resource*.

Figure 6 illustrates the storage mapping for an aggregate disk resource and its constituent resources, shared across two nodes in an RSCT peer domain.

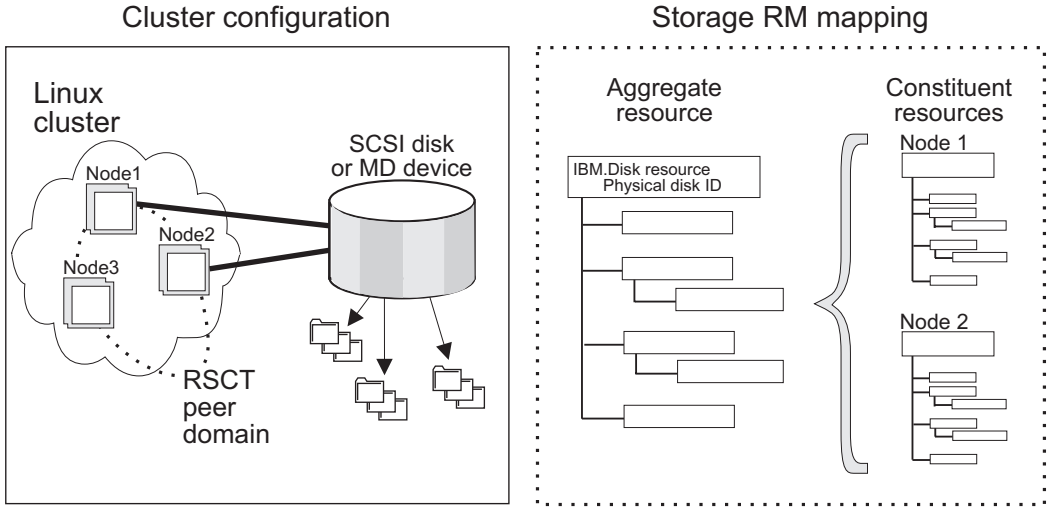

Figure 6. Storage mapping for an aggregate disk resource and its constituent resources

Having an aggregate resource simplifies administration, as you can issue commands against the aggregate resource to affect its constituents. This allows you to manage global resources more efficiently since you can invoke commands against the aggregate resource and the storage resource manager will propagate the changes to the constituent resources. For example, as described in "Configure mount point on harvested IBM.AgFileSystem resources" on page 256, you can set the MountPoint attribute of an AgFileSystem resource to specify the location at which the storage resource manager will attempt to mount the file system. For a global file system resource, you can modify the MountPoint resource attribute of the aggregate IBM.AgFileSystem resource and the storage resource manager will propagate the change to all of the constituent resources of the aggregate resource. This approach to mapping global resources allows you to manage the resource as one shared resource (using the aggregate mapping) or as a single resource on one node (using the constituent mapping for a particular node).

On the other hand, if the storage resource manager determines that a physical disk is attached to a single node, no aggregate will be created. The IBM.Disk resource (as well as each of its associated storage resources) is a *single-fixed resource*.

To summarize, an instance of any of the storage resource manager's resource classes can be:

- a fixed resource (specific to a particular node). The fixed resource is either:
  - a single-fixed resource (when a physical disk is attached to that node only)
  - a constituent resource of an aggregate resource (when a physical disk can be accessed by other nodes)

- an aggregate resource (a global representation of the all the constituent resources that represent the same storage entity)

Attributes of the storage resource manager's resource classes enable you to identify if a particular resource is a single-fixed, constituent, or aggregate resource. See "Determining whether a resource is a single-fixed, constituent, or aggregate resource" on page 259 for more information.

In addition to harvesting storage information when a peer domain is brought online, the storage resource manager repeats the harvest process at regular intervals in order to detect changes in the peer domain's storage configuration (for instance, newly-attached devices, newly-formatted partitions, or the removal or a storage entity). You can set the interval at which storage resources are harvested and can also force immediate harvesting, as described in "Configuring and controlling storage resource harvesting" on page 252.

When a previously harvested resource is not detected by a subsequent harvest operation, it will not, by default, be deleted. Instead, the storage resource manager will set the GhostDevice attribute to 1 to indicate that it is a ghost resource. A ghost resource identifies a storage resource manager resource that might no longer represent an actual storage entity. Instead of being deleted, the resource is, by default, marked as a ghost resource because the storage entity it represents may only be temporarily unavailable. For example, if a previously-harvested disk resource is no longer available to a node, the IBM.Disk resource used to represent the disk will be marked as a ghost resource (GhostDevice=1). This could indicate that the physical disk was removed or it could indicate that the disk is only temporarily unavailable. If the storage resource manager does detect the physical disk during a subsequent harvest operation, it will set the IBM.Disk resource's GhostDevice attribute to 0 to indicate that it is no longer a ghost resource. You can change this default behavior so that previously-harvested storage resources that are not detected by subsequent harvest operations will be automatically deleted rather than marked as ghost resources. See "Enabling automatic deletion of harvested resources when they are not detected by subsequent harvest operations" on page 254 for more information.

"Listing resource information" on page 130 describes how you can use the **lsrsrc** command to list attribute values and the **lsrsrcdef** command to list attribute definitions. You can use these commands to obtain more information about the storage resource manager resource classes. For complete syntax information on these commands, see their online man pages. For detailed information, see also the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

How the storage resource manager monitors the availability of shared storage

If a peer domain is a shared storage environment, multiple nodes within the peer domain have access to the same disk subsystem and, therefore, access to the same disks contained within the disk subsystem.

When there is a disk resource that is attached to multiple nodes in the peer domain, the data integrity of the storage could be compromised if more that one node were to access the disk simultaneously. To avoid this, the storage resource manager's default behavior is to ensure serial access to shared resources using a system of disk reservation and release (implemented using SCSI reserves/releases). Where SCSI reserve capability is not available, as is the case with Linux MD **IBM.Disk** resources and AIX SDD-based multipath disks, the storage resource manager ensures serial access by allowing file systems to be online on only one node in a cluster at a given time. This default behavior can be overridden for particular disks by setting the **DeviceLockMode** attribute of the **IBM.Disk** resource (on Linux) or the IBM.VolumeGroup (on AIX or Linux) to 0 (disabled). See "Disabling disk locking" on page 254.

DeviceLockMode persistent attribute for the IBM.Disk class

The storage resource manager supports the small computer system interface persistent reservation (SCSIPR) disk protection method. The storage resource manager also administers the **DeviceLockMode** attribute for the **IBM.Disk** class. The valid values for the **DeviceLockMode** persistent class attribute are listed in the following table:

Table 69. DeviceLockMode values for persistent class attribute

DeviceLockMode value	Description
0	Device locking is disabled and no protection method is enabled for the **IBM.Disk** class.
1	Device locking for the **IBM.Disk** class is set to the default protection method, that is, SCSI-2.
2	Device locking for the **IBM.Disk** class is set to the SCSIPR protection method.

DeviceLockMode persistent attribute for IBM.Disk resource

You can change the value of the **DeviceLockMode** attribute of an **IBM.Disk** resource to enable new protection methods. You can also set this attribute to follow the **DeviceLockMode** attribute value of the **IBM.Disk** class. The valid values for the **DeviceLockMode** persistent resource attribute are listed in the following table:

Table 70. DeviceLockMode values for persistent resource attribute

DeviceLockMode value	Description
0	Device locking for the **IBM.Disk** resource is disabled and no protection method is enabled.
1	Device locking for the **IBM.Disk** resource is set to the default protection method, that is, SCSI-2.
2	Device locking for the **IBM.Disk** resource is set to the SCSIPR protection method.
3	Device locking for the **IBM.Disk** resource is set to the value that is specified at the class level. This value can be set only for a resource attribute.

Examples

1. To change the device locking for a disk that has physical volume ID (PVID) `00f6a9075f159b3a` to SCSI-2 protection method, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 chrsrc –s 'PVID="00f6a9075f159b3a"' IBM.Disk DeviceLockMode=1
   ```

2. To set the **DeviceLockMode** attribute for the **IBM.Disk** class to SCSIPR, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 chrsrc –c IBM.Disk DeviceLockMode=2
   ```

3. To set the **DeviceLockMode** attribute for a disk `/dev/sdb` to follow the **DeviceLockMode** attribute that is set for the **IBM.Disk** class attribute, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 chrsrc –s 'DeviceName="/dev/sdb"' IBM.Disk DeviceLockMode=3
   ```

   In this example, the **DeviceLockMode** attribute value is set to 3. This means that the device locking attribute is set to the value that is specified at the class level. In this case, the storage resource manager refers to the **DeviceLockMode** class attribute's current value for the disk `/dev/sdb`. For example, if the **DeviceLockMode** class attribute is set to 1, the `/dev/sdb` disk is protected through SCSI-2 reservations.

Resources defined by resource managers can be brought *online* using the generic RMC command **startrsrc** or by an RMC API application calling one of the **mc_online_*** subroutines. The resources can also be taken *offline* using the generic RMC command **stoprsrc** or by an RMC API application calling the one of the **mc_offline_*** subroutines. For each resource class, the terms *online* and *offline* have unique meanings depending on the storage entity that the resource class represents. For the IBM.Disk resource class, for example, the online operation reserves the disk and the offline operation releases it. For the IBM.AgFileSystem resource class, the online operation mounts the file system and the offline operation unmounts it. For the IBM.VolumeGroup resource class, the online operation activates the volume group and the offline operation deactivates it.

Once online, the storage resource manager checks the status of a resource, at intervals determined by the resource class' MonitorInterval property, to make sure the resource is still online.

- For the IBM.Disk resource class, for example, this means passively re-reserving the disk. In cases where the IBM.Disk resource cannot be reserved, such as with Linux MD RAID devices and AIX multipath

disks, disk monitoring will not involve passive re-reserving. Instead, the storage resource manager uses other techniques to verify that the disk is still online, such as ensuring that the hosting operating system (AIX or Linux) still has access to the disk.

- For the IBM.AgFileSystem resource class, this means that means verifying that the file system is available for write operations by writing a hidden StorageRM file.
- For the IBM.VolumeGroup resource class, this means checking that the volume group is still active.

The interval at which resources are monitored is specified separately for each type of resource and can be modified by setting the MonitorInterval property for the resource class. The OpState dynamic attribute of a resource indicates its operation state and can be monitored by an application using the RMC API or from the command line. For information on resource monitoring using the RMC API, see the *RSCT: RMC Programming Guide*. For information on resource monitoring using the command line, see "Monitoring resources using RMC and resource managers" on page 126.

To understand how the storage resource manager protects the integrity of shared resource data, it is important to know that online operations on any storage resources associated with an IBM.Disk resource will, provided that disk locking has not been disabled, also result in the IBM.Disk resource being brought online (in other words, reserved using SCSI reserves, where SCSI reserves are available). So, for example, activating a volume group (IBM.VolumeGroup resource) using the **startrsrc** command or the **mc_online_*** subroutines will first reserve the disk (online the IBM.Disk resource) on which the volume group depends. Similarly, on non-MD Linux systems, mounting a file system represented by a harvested IBM.FileSystem resource using the **startrsrc** command or the **mc_online_*** subroutines will only be successful if the associated disk is (or can be) reserved, where applicable, by the node (and only if the associated volume group is (or can be) activated). The exception to this rule includes MD devices on Linux, where SCSI reservations are not performed, and on AIX, where SCSI disk reservations are attempted, but not required for a volume group to be brought online. This behavior is useful in cases where an AIX volume group contains multiple physical volumes holding mirrored logical volumes, and allows a volume group (and its file systems) to function if some of its disks have failed.

Table 71 describes the online and offline operations for the resource classes available on Linux nodes using disk partitions and shows how disk locking is implemented by bringing container resources online recursively so that no online operation is successful unless the storage resource manager is able to reserve (or otherwise bring online, in the case of non-SCSI disks) the underlying disk resource. It also shows how the online resources are monitored to ensure they are still online. This table assumes that device locking for the IBM.Disk resource has not been disabled (as described in "Disabling disk locking" on page 254).

Table 71. Summary of disk reservation and release behavior on Linux nodes using disk partitions

For resources of this resource class...	The online operation will...	Once online, the storage resource manager, at intervals determined by the resource class' MonitorInterval property, will...	The offline operation will...
IBM.Disk	Reserve the disk, provided that it is not already reserved by another node.	Passively re-reserve the disk.	Release the disk, provided that none of the IBM.Partition or IBM.AgFileSystem resources that it contains are online.
IBM.Partition	Perform the online operation on the associated disk, provided that the associated disk is not reserved by another node. If the online operation on the IBM.Disk resource is successful (the disk is reserved) the IBM.Partition resource is considered online.	Do nothing, since you cannot set the MonitorInterval attribute for the IBM.Partition resource class, and it is disabled (MonitorInterval=0) by default. Since the online status of an IBM.Partition resource is entirely dependent on the IBM.Disk resource that contains it, the storage resource manager does not monitor IBM.Partition resources.	Cause the IBM.Partition resource will be considered offline, provided that no IBM.AgFileSystem resource associated with the IBM.Partition resource is online. Will also perform the offline operation on the IBM.Disk resource, provided that no other IBM.Partition resource associated with the IBM.Disk resource is online.

Table 71. Summary of disk reservation and release behavior on Linux nodes using disk partitions (continued)

For resources of this resource class...	The online operation will...	Once online, the storage resource manager, at intervals determined by the resource class' MonitorInterval property, will...	The offline operation will...
IBM.AgFileSystem	Perform the online operation on the IBM.Partition resource that contains the IBM.AgFileSystem resource and, if the online operation on the IBM.Partition resource is successful, attempt to mount the file system.	Check the status of the file system to ensure it is still mounted.	Unmount the file system. Will also perform the offline operation on the IBM.Partition resource, provided that no other IBM.AgFileSystem resource associated with the IBM.Partition resource is online.

Table 71 on page 245 assumes that the DeviceLockMode attribute of the IBM.Disk resource is set to the default (DeviceLockMode=1).

If device locking is disabled for a disk (DeviceLockMode=0), then it will not be reserved or monitored, unless the DeviceType is 2 (Linux MD RAID) or 4 (AIX vpath). In these latter cases, monitoring is still performed, even though the value of the DeviceLockMode attribute is 0. You cannot change the value of the DeviceLockMode attribute for Linux MD or AIX vpath disk resources. For other types of disks where the DeviceLockMode has been manually set to 0, you can still use the online operation on the IBM.AgFileSystem resource to mount the file system but the storage resource manager will not reserve the underlying disk.

If you disable the use of reserves on a device by setting its DeviceLockMode attribute to 0, then it is your responsibility to ensure that the device is available on the node prior to mounting a dependent file system through the storage resource manager. Because no reserves will be issued by the storage resource manager and only a mount will be issued, the device must be known to be available beforehand.

Table 72 describes the online and offline operations for the resource classes available on AIX or Linux nodes using volume groups, and shows how disk locking is implemented by bringing container resources online recursively so that no online operation is successful unless the storage resource manager is able to reserve the underlying disk resource. It also shows how the online resources are monitored to ensure they are still online. This table assumes that device locking for the IBM.VolumeGroup resource has not been disabled (as described in "Disabling disk locking" on page 254).

Table 72. Summary of disk reservation and release behavior on AIX or Linux nodes using volume groups

For resources of this resource class...	The online operation will...	Once online, the storage resource manager, at intervals determined by the resource class' MonitorInterval property, will...	The offline operation will...
IBM.Disk	Reserve the disk, provided that it is not already reserved by another node and it is a device that supports SCSI-2 reservations. Otherwise, the action to bring the disk resource online depends on the particulars of the device type that the disk resource represents.	Passively re-reserve the disk, where possible. Otherwise, monitoring is performed by verifying the continued access by the operating system to the underlying disk device.	Release the disk, provided that the disk was reserved and that no IBM.Partition or IBM.AgFileSystem resources that it contains are online. If a SCSI reservation was not obtained because the disk does not support them, then the mechanism that is used to bring the disk device offline depends on the type of disk device being used.

Table 72. Summary of disk reservation and release behavior on AIX or Linux nodes using volume groups (continued)

For resources of this resource class...	The online operation will...	Once online, the storage resource manager, at intervals determined by the resource class' MonitorInterval property, will...	The offline operation will...
IBM.VolumeGroup	Perform the online operation on the IBM.Disk resources on which the volume group depends. Linux: If the online operation is successful for all of the IBM.Disk resources, will activate the volume group. AIX: IBM.Disk resources are brought online when possible; ultimate authority to bring up the volume group rests with the AIX varyon logic. This allows volume groups with mirrored logical volumes to function with some disks not available.	Check that the volume group is active.	Deactivate the volume group, provided that no IBM.LogicalVolume resource associated with the IBM.VolumeGroup is online. Will also perform the offline operation on the IBM.Disk resources on which the volume group depends.
IBM.LogicalVolume	Perform the online operation on the IBM.VolumeGroup resource that contains the logical volume. If the online operation on the IBM.VolumeGroup resource is successful (the volume group is activated) the IBM.LogicalVolume resource is considered online.	Do nothing, since you cannot set the MonitorInterval attribute for the IBM.LogicalVolume resource class, and it is disabled (MonitorInterval=0) by default. Since the online status of an IBM.LogicalVolume resource is dependent on the IBM.VolumeGroup resource that contains it, the storage resource manager does not monitor IBM.LogicalVolume resources.	Cause the IBM.LogicalVolume resource will be considered offline, provided that no IBM.AgFileSystem resource associated with the IBM.LogicalVolume resource is online. Will also perform the offline operation on the IBM.VolumeGroup resource, provided that no other IBM.LogicalVolume resource associated with the IBM.VolumeGroup resource is online.
IBM.AgFileSystem	Perform the online operation on the IBM.LogicalVolume resource that contains the IBM.AgFileSystem resource and, if the online operation on the IBM.LogicalVolume resource is successful, attempt to mount the file system.	Check the status of the file system to ensure it is still mounted.	Unmount the file system. Will also perform the offline operation on the IBM.LogicalVolume resource, provided that no other IBM.AgFileSystem resource associated with the IBM.LogicalVolume resource is online.

Table 72 on page 246 assumes that the DeviceLockMode attribute of the IBM.VolumeGroup resource is set to the default (DeviceLockMode=1). If device locking is disabled for the volume group (DeviceLockMode=0), then the underlying physical volume will not be reserved or monitored. You can still use the online operation on the IBM.VolumeGroup and IBM.LogicalVolume resources to activate the volume group, and on the IBM.AgFileSystem resource to mount the file system. The difference is that the storage resource manager will not reserve the underlying disk, and the volume group will be activated without SCSI reserves. If you disable the use of reserves on a device by setting its DeviceLockMode attribute to 0, then it is your responsibility to ensure that the device is available on the node prior to mounting a dependent file system through the storage resource manager. Because no reserves will be issued by the storage resource manager and only a `varyonvg -u` and a mount will be issued, the device must be known to be available beforehand. On AIX, you can use the **lquerypv** command to verify that a physical volume is available on a node.

DeviceLockMode attribute for IBM.VolumeGroup class

The storage resource manager supports the **DeviceLockMode** attribute for the IBM.VolumeGroup class. The valid values for the IBM.VolumeGroup **DeviceLockMode** attribute are listed in the following table.

Table 73. DeviceLockMode attribute values for IBM.VolumeGroup class

DeviceLockMode value	Description
0	When a volume-group is online on a node and another node attempts a **varyonvg** operation on the volume-group, storage resource manager does not protect data.
1	When a volume-group is online on a node and another node attempts a **varyonvg** operation on the volume-group, storage resource manager protects data and the `varyon` operation on that node fails.
2	This is the default value and it is applicable only for resource attributes. Volume-group follows the **DeviceLockMode** class attribute. If a volume-group is active on a node and the **DeviceLockMode** class attribute is 0, the storage resource manager does not provide any protection for the volume-group. In this case, the node that is requesting activation of the volume-group succeeds. If the **DeviceLockMode** class attribute is 1, activation request from other nodes is rejected.

Examples

1. To enable protection from the **varyonvg** attempts on volume group `vg1` by other nodes, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 chrsrc –s 'VGName="vg1"' IBM.VolumeGroup DeviceLockMode=1
   ```

2. To set the **DeviceLockMode** attribute for a volume group `vg1` to follow the **DeviceLockMode** attribute for the IBM.VolumeGroup class, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 chrsrc –s 'VGName="vg1"' IBM.VolumeGroup DeviceLockMode=2
   ```

   In this case, the `vg1` volume group refers to the current value of the **DeviceLockMode** class attribute.

AutoMonitor attribute for IBM.VolumeGroup resource

The **AutoMonitor** attribute can be configured for the IBM.VolumeGroup resources to enable monitoring of the varyon and varyoff operations on the volume groups. The valid values of the **AutoMonitor** attribute are described in the following table:

Table 74. AutoMonitor attribute values for IBM.VolumeGroup resources

AutoMonitor value	Description
0	Varyon and varyoff status monitoring on the volume groups is disabled. The storage resource manager does not reflect the latest state change of the volume group.
1	Varyon and varyoff status monitoring on the volume groups is enabled. The storage resource manager reflects any state change of the volume group. For example, if the volume group is active or inactive externally, the storage resource manager reflects the latest state.
2	Volume group status monitoring is set to follow the current value of the **AutoMonitor** attribute that is set for the IBM.VolumeGroup class. This value is applicable only for resource attribute.

Examples

1. To view the **AutoMonitor** attribute for a volume group resource, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 lsrsrc –s "selectionString" IBM.VolumeGroup Name AutoMonitor
   ```

2. To view the current value of the **AutoMonitor** class attribute, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 lsrsrc –c IBM.VolumeGroup AutoMonitor
   ```

3. To set a volume group `vg1` to follow the **AutoMonitor** class attribute, enter the following command:

   ```
   CT_MANAGEMENT_SCOPE=2 chrsrc –s 'VGName="vg1" ' IBM.VolumeGroup AutoMonitor=2
   ```

   When you run this command, if the **AutoMonitor** class attribute is 0, the `vg1` volume group is not monitored for state changes. If the **AutoMonitor** class attribute is 1, the `vg1` volume group is automatically monitored for state changes.

4. To enable the status monitoring of the varyon and varyoff operations on the volume groups at the class level, enter the following command:

```
CT_MANAGEMENT_SCOPE=2 chrsrc –c IBM.VolumeGroup AutoMonitor=1
```

When you run this command, all the volume groups that follow this class attribute have their effective **AutoMonitor** attribute set to 1.

5. To enable the status monitoring of the varyon and varyoff operations on a volume group resource, enter the following command:

```
CT_MANAGEMENT_SCOPE=2 chrsrc –s "SelectionString" IBM.VolumeGroup AutoMonitor=1
```

VGStateInfo attribute for IBM.AgFileSystem resource

The **VGStateInfo** attribute, which is a read-only dynamic attribute, is a string concatenation of the **OpState** attribute of the IBM.AgFileSystem container volume group that is followed by a blank space and the volume group name. For example, if the IBM.AgFileSystem resource `AgFS1` belongs to a logical volume that is part of a volume group `VG1`, and if `VG1` is inactive (that is, volume-group's **OpState**=2), the **VGStateInfo** attribute value is `2 VG1`.

Example

- To view the **VGStateInfo** attribute, enter the following command:

```
lsrsrc IBM.AgFileSystem VGStateInfo
```

Trace summary

The `trace.summary` file is used to store the trace of the storage resource manager.

To view the contents of the `trace.summary` file through a file `readableSummary`, enter the following command:

```
rpttr /var/ct/RPD/log/mc/IBM.StorageRM/trace.summary > readableSummary
```

where *RPD* is the name of currently active peer domain.

Storage resource manager requirements

Before using the storage resource manager, make sure that your system meets the prerequisites and that you perform the necessary configuration.

Before using the storage resource manager, make sure that your system meets the prerequisites described in "Storage resource manager prerequisites," and that you perform the necessary configuration described in "Storage resource manager configuration requirements" on page 250.

For details about the interoperability of the storage resource manager with various operating systems, server platforms, and disk storage systems, see "Storage resource manager interoperability" on page 263.

Storage resource manager prerequisites

Certain software, hardware, and configuration requirements apply when exploiting storage resource manager functions.

The following software, hardware, and configuration requirements apply for exploiting storage resource manager functions:

- The following non-SCSI devices are supported:
  - On Linux, MD RAID devices
  - On AIX, vpath devices supported by the SDD driver
- When a physical disk is attached to multiple nodes, it must be attached at the same logical unit address on each node.

On Linux nodes, note that:

- Linux Kernel Version 2.4 or 2.6 is required

- Failover versions of HBA drivers are not supported for connecting to DS4000 series disk systems. Instead of using a failover version of an HBA driver, you must use the Redundant Disk Array Controller (RDAC) driver in order to use SCSI-2 reserves, or you can use the DM-MP driver provided by RedHat or SuSE. When RDAC is installed on a node, RDAC must also be installed on all nodes that share any storage with that node.

  To obtain the RDAC driver:

  1. Using a web browser, open the following URL:

     http://www.ibm.com/servers/storage/support/disk/

     This page offers support for disk storage systems.
  2. Click on the name of the appropriate DS4000 series system.

     This opens a support page for the specific DS4000 series system.
  3. On the web page that you have opened, click on the **Download** tab.

     This shows the fixes and drivers that are available for download.
  4. In the list of fixes and drivers, click on the link for **Storage Manager, firmware, HBA and tools (including readmes)**.

     This opens a page for Storage Manager, firmware, HBA, and tool downloads for the DS4000 series system.
  5. On the web page that you have opened, click on the **Storage Mgr** tab.

     This shows the downloadable files for the IBM DS4000 Storage Manager.
  6. Click on the link for the IBM TotalStorage DS4000 Linux RDAC (for the Linux Kernel Version 2.4 or 2.6 as appropriate).

     This opens the download page for the RDAC driver.

On AIX nodes, the AIX MPIO device driver should be used for all DS3000 and DS4000 series disk systems. The required AIX fileset is:

- devices.common.IBM.mpio.rte

Storage resource manager configuration requirements

While the storage resource manager will harvest (detect and collect) the information it needs to create the IBM.Disk, IBM.Partition, and IBM.AgFileSystem resources to represent the disks, configured partitions, and file systems on a Linux disk partition or on an AIX or Linux logical volume, some configuration is necessary for the storage resource manager to work effectively.

In particular, you must do the following:

- On AIX nodes, ensure that the operating system does not automatically vary on the volume group at system restart. This is further described in "Configuring shared volume groups for storage resource manager on AIX" on page 251.
- On AIX nodes, configure file systems to ensure that they are not automatically mounted at system restart. This is further described in "Configuring file systems for storage resource manager on AIX" on page 252.
- In Linux, unless VGs are configured on the local disks, they should not be auto-activated at boot time. To change this behavior, if you have SLES, edit */etc/sysconfig/lvm* and set as follows:

  ```
  LVM_VGS_ACTIVATED_ON_BOOT="@none"
  ```

  If you have */etc/lvm/lvm.conf*, comment out the volume_list variable as follows:

  ```
  # volume_list = [ "vg1", "vg2/lvol1", "@tag1", "@*" ]
  ```

Storage resource manager configuration

Some configuration is needed for file systems and shared volume groups to work with the storage resource manager.

Before you proceed, be sure to review the information in “Storage resource manager requirements” on page 249.

Configuring shared volume groups for storage resource manager on AIX

The storage resource manager can reserve resources on a node and then monitor the reservation of the device on that node. Since there is a natural dependency between a volume group and the physical volumes that comprise the volume group, the action of reserving a volume group implies that the storage resource manager will, where they are available, also reserve the physical volumes, not including Linux MD or AIX vpath IBM.Disk resources, where SCSI reservations are not made.

It is important to specify that the operating system should not automatically vary online the volume group during a system restart. This is important as it prevents a rebooting node from attempting to vary online a disk that may be reserved on another node.

Example: The following **mkvg** command creates a volume group named *storagerm* comprised of only the physical disk *hdisk2*. In this example:

- The **-d** flag restricts the volume group to a maximum of one physical volume.
- The **-n** flag specifies that the operating system should not automatically vary on the volume group during a system restart.

```
mkvg -n -d 1 -y storagerm hdisk2
```

If you do not specify the **-n** flag on the **mkvg** command when creating the volume group, you can later use the **chvg** command with its **-a** flag to specify that the volume group should not be automatically activated during a system restart. For instance:

```
chvg -a n storagerm
```

Once a volume group is configured on one node, it must be imported by any other node that will require shared access to the volume group. When importing a volume group, be aware that, on each node, a different physical volume name may be associated with the same physical volume ID. On each node, you must make sure that the physical volume name specified when importing the volume group is associated with the correct physical volume ID.

Do the following to import a volume group:

1. On the node where the volume group was configured, use the **getlvodm** command to obtain the physical volume ID.

   Example: To obtain the physical volume ID of the physical volume *hdisk2*, enter the command:

   ```
   getlvodm -p hdisk2
   ```

   Result: Output will be the physical volume ID. For instance:

   ```
   000000623e576668
   ```

2. Do the following on each node that will require shared access to the volume group:
   a. Use the **lspv** command to identify the physical volume name associated with the physical volume ID on the node.

      Example:

      ```
      lspv
      ```

      Result: The following output is displayed:

      ```
      hdisk0          000000150f450a54          rootvg          active
      hdisk1          000000623e576668          none
      ```

      In this example, the name associated with the physical volume ID is *hdisk1*.

   b. Use the **importvg** command with its **-n** option to import the volume group. The **-n** option causes the volume group to not be varied online upon the completion of the volume group import into the system.

Example:

```
importvg -y storagerm -n hdisk1
```

c. When you issue the **importvg** command, the AUTO ON value is set to the volume group default value. Use the **chvg** command with its **-a** flag to ensure that the volume group will not be automatically activated during system restart.

```
chvg -a n storagerm
```

Configuring file systems for storage resource manager on AIX

To prevent a rebooting node from attempting to mount a particular file system that may be reserved on another node, you must, on AIX nodes, specify that the file system should not be automatically mounted at system restart. You can specify this using the **-A no** option on either the **crfs** command (when creating the file system) or the **chfs** command (to modify a file system that has already been created).

For example, the following **crfs** command creates a JFS file system on logical volume */dev/lv00* with mount point */storagerm*. The **-A no** option specifies that the file system is not mounted at system startup.

```
crfs -v jfs -d /dev/lv00 -a size=8192 -m /storagerm -A no
```

If the file system has already been created without the **-A no** option specified, you could modify it using the following **chfs** command.

```
chfs -A no /storagerm
```

If you are using a GPFS™ file system, create it using the option to specify that it is not automatically mounted. For example, the following command creates a GPFS file system:

```
mmcrfs
```

Use the option **-A no** within it to specify that it is not automatically mounted as follows:

```
mmcrfs foogpfs "vpath10nsd;vpath9nsd" -A no -D posix -k posix -T /poslmm
```

GPFS also has a command called **mmchfs** which operates similarly to **chfs**. Use it, specifying **-A no**, to change an existing GPFS file system from being auto-mounted to not being auto-mounted.

Although the preceding commands will prevent a rebooting node from automatically mounting the file system, you must still use the **chrsrc** command to identify the mount point to the storage resource manager (as described in "Configure mount point on harvested IBM.AgFileSystem resources" on page 256).

Storage resource manager optional configuration

Optionally, you can configure the storage resource manager to perform several operations.

You can optionally configure the storage resource manager to:

- Determine certain aspects of the resource harvesting process such as the interval at which harvesting occurs. You can also force immediate storage resource harvesting.
- Disable the storage resource manager's system of disk reservation and release (implemented using SCSI reserves/releases) when synchronized access to shared resources is not needed.
- Control the way file systems are mounted.

Configuring and controlling storage resource harvesting

The storage resource manager automatically harvests information about physical disks (and related storage entities) and represents this information using its resource classes.

This is described in "How the storage resource manager gathers and represents storage data" on page 240, The storage resource manager will harvest storage resources when a peer domain is brought online and then at regular intervals to detect changes in the storage configuration (such as newly attached or removed devices). You can:

- Determine how often the storage resource manager will scan for changes in the storage configuration by setting the HarvestInterval persistent resource attribute of the IBM.Disk resource class.
- Refresh the storage configuration at any time by using the **refrsrc** command on the IBM.Disk resource class.
- Enable automatic deletion of harvested resources when they are not detected by subsequent harvest operations.

Setting the interval at which storage resources are harvested:

After its initial resource harvesting when the peer domain is brought online, the storage resource manager will then repeat the harvest operation at set intervals to detect changes in the storage configuration.

This interval is determined by the value of the HarvestInterval persistent resource attribute of the IBM.Disk resource class. You can obtain the current value of HarvestInterval persistent resource attribute using the **lsrsrc** command, and can set it using the **chrsrc** command.

The following **lsrsrc** command displays the current harvest interval value. The value is always expressed in seconds.

```
lsrsrc -c IBM.Disk HarvestInterval
```

The following command output is displayed:

```
Resource Class Persistent Attributes for IBM.Disk
resource 1:
    HarvestInterval  = 7200
```

The following **chrsrc** command modifies the harvest interval value. When modifying the HarvestInterval property, be sure to express the value in seconds. The shortest harvest interval you can specify is 60 seconds.

```
chrsrc -c IBM.Disk HarvestInterval=1440
```

To completely disable the periodic resource harvesting, you can set the HarvestInterval to 0:

```
chrsrc -c IBM.Disk HarvestInterval=0
```

If you do disable the periodic resource harvesting, be aware that the storage resource manager will only perform the harvest operation when the peer domain is brought online or when you manually refresh the configuration (as described in "Forcing immediate storage resource harvesting").

Forcing immediate storage resource harvesting:

If you change physical or logical storage entities for a node by adding, removing, or otherwise modifying them (such as placing them in a new location), the storage resource manager will detect these changes during the next scheduled harvest operation (as determined by the HarvestInterval attribute of the IBM.Disk resource class).

You can also force the storage resource manager to immediately perform the harvest operation to pick up the configuration changes. To do this, issue the **refrsrc** command against the IBM.Disk resource class:

```
refrsrc IBM.Disk
```

The preceding command will cause the current node to re-harvest all storage resources on the node, and propagate all configuration updates to the other nodes within the peer domain.

Enabling automatic deletion of harvested resources when they are not detected by subsequent harvest operations:

When the storage resource manager no longer detects a resource it had previously harvested, the storage resource manager will identify the resource as a ghost resource by setting the resource's GhostDevice attribute to 1.

This is described in "How the storage resource manager gathers and represents storage data" on page 240. A ghost resource represents a storage resource manager resource that may no longer represent an actual storage entity. It is marked as a ghost resource because the storage entity it represents may only be temporarily unavailable.

Although the storage resource manager will, by default, mark unavailable resources as ghosts, you can change this behavior by setting the AutoDelete attribute of the IBM.Disk resource class to 1. Setting the AutoDelete resource class attribute to 1 enables the automatic deletion of harvested resources when they are not detected by subsequent harvest operations. To enable automatic deletion of undetected resources, enter the following on a node that is online in the peer domain:

```
chrsrc -c IBM.Disk AutoDelete=1
```

Disabling disk locking

Learn how the storage resource manager uses, by default, a system of disk reservation and release (implemented using SCSI reserves/releases) to ensure synchronized access to shared resources.

"How the storage resource manager monitors the availability of shared storage" on page 243 describes how the storage resource manager uses, by default, a system of disk reservation and release (implemented using SCSI reserves/releases) to ensure synchronized access to shared resources. It also describes the online and offline operations for the various storage resource manager resource classes, and how disk locking is implemented by bringing up container resources recursively.

Disk locking is the default behavior of the storage resource manager for disks that support SCSI-2 reservations. The default behavior is different for other device types, such as Linux MD RAID devices and AIX SDD-managed vpath devices, for which no reservations are done.

The following information only applies to disk resources that support SCSI reservations; their default DeviceLockMode value is always 1. You can override this behavior for individual disks, as follows:

- On Linux nodes using disk partitions, setting the IBM.Disk resource's DeviceLockMode to 0.

  Example: To disable disk locking for the disk *000000150f450a54*, enter the following command on a node that is online in the peer domain:

  ```
  chrsrc -s "Name=='000000150f450a54'" IBM.Disk DeviceLockMode=0
  ```

- On AIX and Linux nodes using logical volumes, setting the IBM.VolumeGroup resource's DeviceLockMode to 0.

  Example: To disable disk locking for volume group *hd2* and its underlying physical volume, enter the following command on a node that is online in the peer domain:

  ```
  chrsrc -s "Name=='hd2'" IBM.VolumeGroup DeviceLockMode=0
  ```

Configuring file system mounting on IBM.AgFileSystem resources

The online operation for IBM.AgFileSystem resources is to mount the file system represented by the IBM.AgFileSystem resource.

This is described in "How the storage resource manager monitors the availability of shared storage" on page 243

If the storage resource manager encounters an error during the initial mount of a file system, it will by default, attempt to resolve the problem by issuing the **fsck** command, and then will attempt another mount. You can disable or modify this behavior.

For harvested resources, the mount point will, by default, be the one specified by LVM on AIX or in the */etc/fstab* file on Linux. You can modify this by setting the IBM.AgFileSystem resource's MountPoint attribute.

As described in "How the storage resource manager monitors the availability of shared storage" on page 243, disk locking is implemented by bringing container resources on recursively so that no online operation is successful unless the storage resource manager is able to reserve the underlying disk resource. Unless this behavior has been disabled, mounting a harvested file system resource will also reserve the underlying disk resource, where the disk resource supports reservation. You can also create instances of the IBM.AgFileSystem resource class that are independent of any of the harvested devices, thus enabling you to mount file systems without requiring interactions with a physical disk.

Configuring method used to resolve problems when initial file system mount fails:

If the storage resource manager encounters an error during the initial mount of a file system, by default it attempts to resolve the problem by issuing the **fsck** command, and then attempts another mount.

The **fsck** command checks for, and repairs, file system inconsistencies. The exact format of the **fsck** command issued by the storage resource manager is different on Linux and AIX machines and can be modified or disabled for either all file systems (by setting the PreOnlineMethod attribute of the IBM.AgFileSystem resource class) or for a particular file system (by setting the **PreOnlineMethod** attribute of the particular **IBM.AgFileSystem** resource).

Table 75 summarizes the **PreOnlineMethod** attribute settings for the **IBM.AgFileSystem** resource class. This setting determines whether the storage resource manager will, by default, use the **fsck** command to check and repair file systems and, if so, the exact format of the **fsck** command.

Table 75. PreOnlineMethod attribute settings for the IBM.AgFileSystem resource class

PreOnlineMethod setting	**File system checking method in Linux**	**File system checking method in AIX**
1	No operation	No operation
2 (default)	**fsck –a** *fs_name*	**fsck -fp** *fs_name*
3	**fsck –a** *fs_name*	**fsck –p –o nologredo** *fs_name*

For example, the following command disables the automatic checking and repair of file systems by setting PreOnlineMethod attribute of the IBM.AgFileSystem resource class to 0. This setting would then be the default for all file systems, but can be overridden by a particular file system.

```
chrsrc -c IBM.AgFileSystem PreOnlineMethod=0
```

The PreOnlineMethod attribute setting of the IBM.AgFileSystem resource class determines the default method for all file systems, but can be overridden for a particular file system by setting the PreOnlineMethod attribute of the individual IBM.AgFileSystem resource. Table 76 summarizes the PreOnlineMethod attribute settings for an IBM.AgFileSystem resource.

Table 76. PreOnlineMethod attribute settings for an IBM.AgFileSystem resource

PreOnlineMethod setting	**File system checking method in Linux**	**File system checking method in AIX**
0 (default)	Use the method determined by the **IBM.AgFileSystem** resource class.	Use the method determined by the **IBM.AgFileSystem** resource class.
1	No operation	No operation
2	Use the following **fsck** command format: **fsck –a** *fs_name*	Use the following **fsck** command format: **fsck -fp** *fs_name*

Table 76. PreOnlineMethod attribute settings for an IBM.AgFileSystem resource (continued)

PreOnlineMethod setting	File system checking method in Linux	File system checking method in AIX
3	Use the following **fsck** command format: **fsck –a** *fs_name*	Use the following **fsck** command format: **fsck –p –o nologredo** *fs_name*

For example, the following command disables the automatic checking and repair of the file system *fs1* by setting its PreOnlineMethod resource attribute to 1.

```
chrsrc -s "Name=='/fs1'" IBM.AgFileSystem PreOnlineMethod=1
```

Table 77 summarizes the PreOfflineMethod attribute settings for the **IBM.AgFileSystem** resource class. On a request of stopping or unmounting a busy **IBM.AgFileSystem** resource class, this setting determines whether the storage resource manager ends the process that is using it.

Table 77. PreOfflineMethod attribute settings for the IBM.AgFileSystem resource class

PreOfflineMethod setting	File-system checking method
1 (default)	No operation
2	Use the following **fsck** command format: **fuser –kc** *MountPoint*

By default, the PreOfflineMethod attribute settings are disabled for the IBM.AgFileSystem class and its resources; however, it can be overridden. To change the PreOfflineMethod attribute settings for all of the resources of the IBM.AgFileSystem class, enter the following command:

```
chrsrc –c IBM.AgFileSystem PreOfflineMethod=2
```

To change PreOfflineMethod for a particular resource, enter the following command:

```
chrsrc –s "Select string" IBM.AgFileSystem PreOfflineMethod=1
```

Table 78. PreOfflineMethod attribute settings for the IBM.AgFileSystem resource

PreOfflineMethod setting	File system checking method
0 (default)	Use the method defined by the **IBM.AgFileSystem** resource class.
1	No operation
2	Use the following **fsck** command format: **fuser –kc** *MountPoint*

Configure mount point on harvested IBM.AgFileSystem resources:

When the storage resource manager harvests file system information, it identifies the file system mount point (as specified by LVM on AIX or in the */etc/fstab* file on Linux) and maps this information to the IBM.AgFileSystem resource's SysMountPoint attribute.

When attempting to mount a harvested IBM.AgFileSystem resource, the storage resource manager will, by default, use the mount point identified on the SysMountPoint attribute. You can override this to specify a different mount point by setting the IBM.AgFileSystem resource's MountPoint attribute. If an IBM.AgFileSystem resource's MountPoint attribute is set, the storage resource manager will use the mount point it specifies instead of the one specified by the SysMountPoint attribute.

If the desired mount point for a file system is the same on each node sharing the file system, then the MountPoint attribute can be set through the file system aggregate resource definition. For example, to set the mount point for the aggregate file system resource *fs1* to */storagermdir*:

1. Set the management scope to RSCT peer domain (a value of 2):

   ```
   export CT_MANAGEMENT_SCOPE=2
   ```

2. Use the generic RMC command **chrsrc** to set the persistent resource attribute MountPoint for the aggregate IBM.AgFileSystem resource.

```
chrsrc -s "Name=='/fs1' && ResourceType==1" \
IBM.AgFileSystem MountPoint=/storagermdir
```

If the desired mount point is different on the various nodes requiring access to the file system, then the MountPoint attribute must be set through each file system constituent resource definition. For example, to set the mount point for the constituent file system resource *fs1*, the node may be specified:

1. Set the management scope to RSCT peer domain (a value of 2):

```
export CT_MANAGEMENT_SCOPE=2
```

2. Use the generic RMC command **chrsrc** to set the persistent resource attribute MountPoint for an IBM.AgFileSystem resource. In this example, the constituent node is *node1* and is identified in a selection string using the NodeNameList persistent resource attribute.

```
chrsrc -s "Name=='fs1' && NodeNameList=='node1'" IBM.AgFileSystem \
MountPoint=/node1strmdir
```

Note: You can also update the SysMountPoint attribute for a resource. To do this, first change the mount point value on the node (by editing the */etc/fstab* file on Linux or by using the **chfs** command on AIX). Once you have changed the mount point value, the storage resource manager will pick up the new value during the next harvest operation. You can instruct the storage resource manager to immediately harvest resources as described in "Forcing immediate storage resource harvesting" on page 253. The SysMountPoint attribute of the IBM.AgFileSystem resource will then reflect the new value.

Defining an IBM.AgFileSystem resource that is independent of harvested resources:

The storage resource manager will automatically *harvest* (detect and collect) information about physical disks and related storage entities within a peer domain and will map these to instances of its resource classes.

This is described in "How the storage resource manager gathers and represents storage data" on page 240. You can also create instances of the IBM.AgFileSystem resource class that are independent of any of the harvested devices. This ability enables you to create IBM.AgFileSystem resources to represent file systems that are not harvested. Such user-defined IBM.AgFileSystem resources are particularly useful for mounting file systems without requiring interactions with a physical disk or partition.

For example, you could create an IBM.AgFileSystem resource instance to represent an NFS file system. Since the IBM.AgFileSystem is user-defined and so has no association with harvested IBM.Disk and related resources, the Storage Resource manager will be able to mount the file system using the mount operation only. No further operations such as reserving a disk or activating a volume group will be required.

A user-defined IBM.AgFileSystem resource can represent the same file system as a harvested IBM.AgFileSystem resource. This would enable you to choose whether to mount the file system while reserving the underlying devices, or to mount the file system without interacting with underlying devices depending on which resource instance is utilized.

To create an IBM.AgFileSystem resource that is independent of the harvested resources, use the **mkrsrc** command. On the **mkrsrc** command, use the:

- Name attribute to assign a name to the IBM.AgFileSystem resource.
- Vfs attribute to specify the type of file system the resource will represent.
- DeviceName attribute to specify the name of an existing file system
- MountPoint attribute to specify an existing mount point.

- Force attribute if an IBM.AgFileSystem resource that specifies the same DeviceName attribute value already exists. This will be the case if the file system was harvested by the storage resource manager. The Force attribute is not needed if the device is not already known to the storage resource manager (as in the case of NFS).

To create a single-fixed resource on just one node of the peer domain, specify ResourceType=0 on the **mkrsrc** command. To create an aggregate resource (with a constituent resource on each node of the peer domain), specify ResourceType=1 and NodeNameList="" on the **mkrsrc** command.

For example, the following **mkrsrc** command creates a single-fixed IBM.AgFileSystem resource on the node where the command is issued.

```
mkrsrc IBM.AgFileSystem Name=/fs1 vfs=jfs Force=1 \
ResourceType=0 DeviceName=/dev/fslv03 MountPoint=/bkp
```

Instead of a single-fixed resource as in the preceding example, the following **mkrsrc** command creates an aggregate IBM.AgFileSystem resource (with a constituent IBM.AgFileSystem resource on each node in the peer domain).

```
mkrsrc IBM.AgFileSystem Name=/fs1 vfs=jfs Force=1 \
ResourceType=1 NodeNameList="" DeviceName=/dev/fslv03 MountPoint=/bkp
```

Listing resource information

To list resource information, use the generic RMC command **lsrsrc** to list the attribute values of IBM.Disk, IBM.VolumeGroup, IBM.LogicalVolume, IBM.Partition, and IBM.AgFileSystem resources.

The names of the listed attributes are often self-explanatory, but you can always get more information on the purpose or meaning of resource attributes by using the **lsrsrcdef** command to list attribute definitions. Both of these commands are described in "Listing resource information" on page 130. Detailed information is also provided in *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Discerning the relationship between storage resources

The storage model for Linux nodes differs from the storage model for AIX nodes. On Linux nodes, disk resources may contain partition resources which contain file system resources. On AIX or Linux nodes, volume group resources (which are dependent on disk resources) contain logical volume resources, which contain file system resources.

Additionally, an instance of the storage resource manager's resource classes may be a single-fixed resource, an aggregate resource, or a constituent resource of an aggregate. As described in "How the storage resource manager gathers and represents storage data" on page 240, when a physical disk is specific to one node only, the resource instances created by the storage resource manager for that node will be single-fixed resources. When a physical disk can be accessed by multiple nodes, however, the fixed resources created by the storage resource manager for those nodes will be constituent resources of a separate aggregate resource that the storage resource manager creates to represent the global view of the common, shared, storage entity. Having the aggregate resource simplifies administration, as you can issue commands against the aggregate to affect its constituents.

You can use the **lsrsrc** command to return attribute values of storage resources to discern the relationship between storage resources. This includes identifying whether a resource is a single-fixed resource, a fixed resource that is a constituent of an aggregate, or an aggregate resource. If the resource is a constituent resource, you can identify its aggregate. If the resource is an aggregate, you can identify its constituents. See "Determining whether a resource is a single-fixed, constituent, or aggregate resource" on page 259 for more information.

You can also use the **lsrsrc** command to discern the relationship between the storage entities that are mapped to the storage resource manager's resource instances. For example, you can identify the partition

that contains a particular file system or the volume group to which a particular logical volume belongs. The storage resource manager harvests both the storage entities and their relationships in terms of containment.

- For IBM.AgFileSystem, IBM.LogicalVolume, and IBM.Partition resources, the relationships among the storage entities are captured by the resource attributes ContainerResource and ContainerResourceID.
- For IBM.Disk resources, the relationships amont the storage entities are captured by the resource attributes DependentResource and DependentResourceId.

Typically, an explicit understanding of the relationships is not required as the storage resource manager handles these relationships internally during any online or offline operation. For example, the online request issued against an instance of the IBM.AgFileSystem resource class will propagate down to the appropriate lower level logical and physical devices. Likewise, an offline operation will first offline the IBM.AgFileSystem resource and then propagate down to the IBM.Disk resource instance. However, you can use the **lsrsrc** command to identify the relationship between storage entities, as described in "Discerning the relationship between storage entities on Linux nodes using disk partitions" on page 260 and "Discerning the relationship between storage entities on AIX or Linux nodes using volume groups" on page 262.

Determining whether a resource is a single-fixed, constituent, or aggregate resource:

A resource's ResourceType attribute indicates whether the resource is a fixed resource or an aggregate resource.

As described in "How the storage resource manager gathers and represents storage data" on page 240, any instance of one of the storage resource manager's resource classes can be a:

- fixed resource (specific to a particular node). The fixed resource is either:
  - a single-fixed resource (when a physical disk is attached to that node only)
  - a constituent resource of an aggregate resource (when a physical disk can be accessed by other nodes)
- an aggregate resource (a global representation of all the constituent resources that represent the same storage entity)

ResourceType value	Resource type
0	A fixed resource (either a single-fixed resource or a constituent resource of an aggregate resource)
1	An aggregate resource

To determine if a fixed resource is a single-fixed resource or a constituent resource of an aggregate resource, see to its AggregateResource attribute value. If a fixed resource is a constituent resource of an aggregate resource, then its AggregateResource attribute will be the resource handle for the aggregate resource. If the resource is instead a single fixed resource or an aggregate resource, then the AggregateResource attribute value will be an invalid or null resource handle.

Example: The following **lsrsrc** command lists the physical disks:

```
lsrsrc IBM.Disk Name ResourceType AggregateResource
```

Result: Command output is similar to the following. The ResourceType and AggregateResource attributes show if the resource is single-fixed, constituent, or aggregate resource. Resource 1 is a single fixed resource, resource 2 is a constituent resource of an aggregate resource, and resource 3 is an aggregate resource.

```
Resource Persistent Attributes for IBM.Disk
      resource 1:
              Name = "00005264d21adb2e"
              ResourceType = 0
              AggregateResource=0x3fff 0xffff 0x00000000 0x00000000 0x00000000 0x
```

```
00000000
      resource 2:
              Name = "000000371e5766b8"
              ResourceType = 0
              AggregateResource=0x2036 0xffff 0x5f47b7ad 0x2b379874 0x8f9cc90c 0x
e738f35b
      resource 3:
              Name = "000069683404ed54"
              ResourceType = 1
              AggregateResource=0x3fff 0xffff 0x00000000 0x00000000 0x00000000 0x
00000000
:
```

Listing aggregate resources:

If a fixed resource is a constituent resource of an aggregate, its AggregateResource attribute value is the resource handle of the aggregate resource.

You can use this resource handle in a selection string on the **lsrsrc** command to list information for the aggregate resource. For example, the following attribute values for an IBM.Disk resource identifies it as a constituent resource of an aggregate resource.

```
resource 2:
       Name = "000000371e5766b8"
       ResourceType = 0
       AggregateResource=0x2036 0xffff 0x5f47b7ad 0x2b379874 0x8f9cc90c 0xe738f35b
```

To list information for the aggregate IBM.Disk resource, you would use the following **lsrsrc** command.

```
lsrsrc -s "ResourceHandle=='0x2036 0xffff 0x5f47b7ad 0x2b379874 0x8f9cc90c \
0xe738f35b'" IBM.Disk
```

To list all instances of a storage resource manager resource class that represent aggregate resources, you can use the ResourceType attribute in a selection string on the **lsrsrc** command. For example, the following command lists all the aggregate IBM.Disk resources for the peer domain. You could use similar commands to list aggregate resources of the other storage resource manager resource classes.

```
lsrsrc -s 'ResourceType==1' IBM.Disk
```

Listing constituent resources of an aggregate resource:

If a fixed resource is a constituent resource of an aggregate, its AggregateResource attribute value is the resource handle of the aggregate resource.

If you know the resource handle of an aggregate resource, you can list all of its constituent resources using the AggregateResource attribute in a selection string on the **lsrsrc** command.

Example: If the resource handle of an aggregate IBM.Disk resource is `0x2036 0xffff 0x5f47b7ad 0x2b379874 0x8f9cc90c 0xe738f35b`, you can list all of its constituent resources by issuing the following **lsrsrc** command:

```
lsrsrc -s "AggregateResource=='0x2036 0xffff 0x5f47b7ad 0x2b379874 0x8f9cc90c \
0xe738f35b'" IBM.Disk
```

You could use similar commands to list constituent resources for aggregates resources of the other Storage resource manager resource classes.

Discerning the relationship between storage entities on Linux nodes using disk partitions:

On Linux nodes, disk resources can contain partition resources, which contain file system resources.

A file system resource (IBM.AgFileSystem) has a ContainerResourceId attribute that enables you to identify the partition resource (IBM.Partition) that contains the file system. Likewise, a partition resource has a ContainerResourceId attribute that enables you to identify the disk resource (IBM.Disk) that contains the partition. To discern the relationship between the particular disk, partition, and file system resources, you can use the **lsrsrc** command to get the value of the ContainerResourceId persistent resource attribute for file system and partition resources.

Example: The following **lsrsrc** command uses a selection string to identify a particular file system named */mnt/sdg2*:

```
lsrsrc -t -s "Name=='/mnt/sdg2' && NodeNameList=='node1'" \
 IBM.AgFileSystem Name ContainerResourceId
```

Result: In the command output, the value of the ContainerResourceId attribute is the resource ID of the partition resource that contains the file system:

```
Resource Persistent and Dynamic Attributes for IBM.AgFileSystem
Name        ContainerResourceId
"/mnt/sdg2" "600a0b80000f013800000011406ec529.2"
```

For partition resources, the ContainerResourceId is the ResourceId of the IBM.Disk resource that contains the partition.

Example: The following **lsrsrc** command uses a selection string to identify the partition resource whose resource ID was returned in the preceding output:

```
lsrsrc -t -s "ResourceId=='600a0b80000f013800000011406ec529.2' && \
 NodeNameList=='node1'" IBM.Partition Name ContainerResourceId
```

Result: In the command output, the value of the ContainerResourceId attribute is the resource ID of the disk resource that contains the partition:

```
Resource Persistent and Dynamic Attributes for IBM.Partition
Name                                  ContainerResourceId
"600a0b80000f013800000011406ec529.2" "600a0b80000f013800000011406ec529"
```

Example: The following command lists information for the IBM.Disk resource which contains the IBM.Partition and IBM.AgFileSystem resources shown above:

```
lsrsrc -s 'ResourceId=="600a0b80000f013800000011406ec529" && \
 NodeNameList=="node1"' IBM.Disk Name DeviceName DeviceInfo OpState
Resource Persistent and Dynamic Attributes for IBM.Disk
resource 1:
        Name       = "600a0b80000f013800000011406ec529"
        DeviceName = "/dev/sdg"
        DeviceInfo = "HOST=4,CHAN=0,ID=0,LUN=5"
        OpState    = 2
```

You can also use the ContainerResource resource attribute to identify containing resources without needing to specify a NodeNameList in the **lsrsrc** query. In the preceding examples, we use the NodeNameList as this is a shared IBM.AgFileSytem resource and, without specifying the NodeNameList, all constituent resources and the aggregate resource would be shown. Another way to do the same as the above without needing to specify the NodeNameList in the select statement would be to use ContainerResource resource attribute. The ContainerResource resource attribute contains the value of the resource handle (a unique ID within the domain) to identify the resource.

All constituent file system and constituent partition resources are contained in constituent disk and constituent partition resources. All aggregate file systems and aggregate partition resources are contained in aggregate file system and aggregate partition resources.

Discerning the relationship between storage entities on AIX or Linux nodes using volume groups:

On AIX or Linux nodes, volume group resources are dependent on one or more disk resources.

The volume group resources contain logical volume resources, which contain file system resources. A file system resource (IBM.AgFileSystem) has a ContainerResourceId attribute that enables you to identify the logical volume resource (IBM.LogicalVolume) that contains the file system. Likewise, the logical volume resource has a ContainerResourceId attribute that enables you to identify the volume group (IBM.VolumeGroup) that contains the logical volume. The relationship between an IBM.VolumeGroup resource and an IBM.Disk resource is not identified on the volume group resource; instead, each IBM.Disk resource contains a DependentResourceId attribute that identifies the volume group to which it belongs.

To discern the relationship between the various storage resources, you can use the **lsrsrc** command to get the value of the ContainerResourceId persistent resource attribute for file system and logical volume resources, and the value of the DependentResourceId persistent resource attribute for disk resources.

Example: The following **lsrsrc** command lists the ContainerResourceId information for all file systems on *node1*:

```
lsrsrc -t -s "NodeNameList=='node1'" IBM.AgFileSystem \
 Name DeviceName ContainerResourceId
```

Result: Command output is similar to the following. For each resource listed, the ContainerResourceId attribute is the ResourceId (logical volume ID) of the logical volume that contains the file system. The DeviceName resource attribute is the associated logical volume device name.

```
Resource Persistent and Dynamic Attributes for IBM.AgFileSystem
Name            DeviceName     ContainerResourceId
"/myfs"         "/dev/lv18"    "0024a09a00004c00000001091bda12ea.3"
"/opt"          "/dev/hd10opt" "0024a09a00004c00000001069d28c253.9"
"/home"         "/dev/hd1"     "0024a09a00004c00000001069d28c253.8"
"/var"          "/dev/hd9var"  "0024a09a00004c00000001069d28c253.6"
"/usr"          "/dev/hd2"     "0024a09a00004c00000001069d28c253.5"
"/"             "/dev/hd4"     "0024a09a00004c00000001069d28c253.4"
"/tmp"          "/dev/hd3"     "0024a09a00004c00000001069d28c253.7"
```

The preceding example uses a selection string to indicate that the **lsrsrc** command should return information only for *node1*. You could also use a selection string to indicate that the **lsrsrc** command should return information only for a particular file system on the node.

Example: The following **lsrsrc** command lists the ContainerResourceId information for the */usr* file system on *node1*:

```
lsrsrc -t -s "Name=='/usr' && NodeNameList=='node1'" IBM.AgFileSystem \
 Name DeviceName ContainerResourceId
```

Result: Command output similar to the following is displayed. The value of the ContainerResourceId is the resource ID (logical volume ID) of the logical volume which contains the file system */usr* on node *node1*.

```
Resource Persistent and Dynamic Attributes for IBM.AgFileSystem
Name   DeviceName ContainerResourceId
"/usr" "/dev/hd2" "0024a09a00004c00000001069d28c253.5"
```

Once you know the resource id (logical volume ID) of the logical volume that contains the file system resource, you can use the **lsrsrc** command to identify the volume group to which the logical volume belongs.

Example: The following **lsrsrc** command lists the ContainerResourceId information for a logical volume ID:

```
lsrsrc -t -s "ResourceId=='0024a09a00004c00000001069d28c253.5' && \
 NodeNameList=='c48f1rp16'" \
 IBM.LogicalVolume Name DeviceName ContainerResourceId
```

Result: Command output is similar to the following. The ContainerResourceId is the volume group ID of the volume group that contains the logical volume.

```
Resource Persistent and Dynamic Attributes for IBM.LogicalVolume
Name                                  DeviceName ContainerResourceId
"0024a09a00004c00000001069d28c253.5" "/dev/hd2" "0024a09a00004c00000001069d28c253"
```

With the volume group ID, you can find the disk associated with the volume group in several ways. One way is to use the DependentResourceId field of IBM.Disk resource class. Given the volume group ID obtained in the previous example, you can list the IBM.Disk resource which is a member of the volume group in question.

Example: The following **lsrsrc** command lists the device information for the IBM.Disk resource that is a member of the specified volume group ID (DependentResourceId):

```
lsrsrc -t -s "DependentResourceId=='0024a09a00004c00000001069d28c253' \
 && NodeNameList=='node1'" \
 IBM.Disk Name DeviceName DeviceInfo
```

Result: Command output is similar to the following:

```
Resource Persistent and Dynamic Attributes for IBM.Disk
Name              DeviceName    DeviceInfo
"0024a09a1b6f2012" "/dev/hdisk0" "VG=rootvg VGID=0024a09a00004c00000001069d28c253"
```

Storage resource manager interoperability

The information in this topic describes storage resource manager interoperability in terms of its support for various operating systems and distributions, server platforms, and disk storage systems.

Supported operating systems, distributions, and server platforms

The storage resource manager supports various operating systems and distributions running on various server platforms.

Table 79 shows the interoperability matrix of operating systems and distributions running on various server platforms that the storage resource manager supports.

Table 79. Operating systems, distributions, and hardware platforms supported by the storage resource manager

Operating system or distribution	BladeCenter (Intel, AMD)	BladeCenter (POWER)	Power Systems servers	System x	System z
IBM AIX 6.1 and 7.1	no	yes	yes	no	no
Red Hat Enterprise Linux 5	yes	yes	yes	yes	yes
SuSE Linux Enterprise Server 10	yes	yes	yes	yes	yes
SuSE Linux Enterprise Server 11	yes	yes	yes	yes	yes

Currently, for the IBM System x platform, the supported architecture is x86 (32-bit and 64-bit). See “Supported Linux distributions for RSCT” on page 31 for more information.

Tested IBM disk storage systems

A number of IBM disk storage systems have been tested with the RSCT storage resource manager running on the supported operating systems and distributions.

Note: RSCT does not support a Trusted AIX environment.

Table 80 lists the IBM disk storage systems that were tested with the RSCT storage resource manager running on the supported operating systems and distributions.

Table 80. IBM System Storage and TotalStorage disk systems tested with the storage resource manager

Operating system or distribution	DS3400	DS4000	DS5000, DS5020	DS6000	DS8000	SAN Volume Controller	Enterprise Storage Server® (ESS)	Notes
AIX 6.1 and 7.1	no	yes	yes	no	yes	yes	no	1, 2, 4, 5, 7
RHEL 5	no	yes	no	no	yes	no	no	3, 6
SLES 10	no	yes	no	no	no	no	no	
SLES 11	no	yes	no	no	no	no	no	

Notes:
1. Testing was performed on IBM System Storage® DS4400 equipment.
2. Testing was performed on IBM SAN Volume Controller (SVC) 4.1.1.535.
3. Testing was performed on IBM System Storage DS8000 equipment.
4. Tested with the IBM System Storage Multipath SDD driver.
5. Tested with the IBM System Storage Multipath SDDPCM driver.
6. Tested with the QLogic driver supplied on the Red Hat Enterprise Linux 5 AS distribution media.
7. Testing was performed on IBM System Storage DS5300 equipment.

Supported storage devices and file systems

RSCT storage resource manager support is available for various systems and devices.

The following information describes the RSCT storage resource manager support for:

- Single-path and multipath storage devices of the IBM System Storage DS4000 family
- Single-path and multipath storage devices on other IBM System Storage disk systems
- Single-path, non-DS4000 storage devices
- Network file systems (NFS)

Note: Although the storage resource manager can harvest and monitor any number of supported, attached, external storage devices, it will not harvest more than one local disk on each system.

Support for single-path and multipath storage devices of the IBM System Storage DS4000 family

Table 81 on page 265 describes the storage resource manager support for single-path and multipath storage devices of the IBM System Storage DS4000 family.

Table 81. Storage resource manager support for single-path and multipath storage devices of the IBM System Storage DS4000 family

AIX	Linux on the System z hardware platform	Power Systems running on Linux Linux on the System x hardware platform
Full support is available for single-path I/O (SPIO) storage devices and for multipath I/O (MPIO) storage devices: • Harvested IBM.AgFileSystem resources can be automated. IBM.AgFileSystem resources are harvested if they are of type *jfs* or *jfs2* and reside on storage entities that are harvested themselves (storage entities of classes IBM.LogicalVolume, IBM.VolumeGroup, IBM.Disk). • User-defined IBM.AgFileSystem resources can be automated (for example, network file systems). • SCSI-2 reservation is supported for single-path devices and for multipath devices using the Redundant Disk Array Controller (RDAC) driver. **Limitations:** • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster.	Only user-defined IBM.AgFileSystem resources can be automated. **Limitations:** • Resource harvesting is not supported. Even if harvesting is successful, the harvested resources cannot be automated. • SCSI reservation is not supported. • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster.	Full support is available for SPIO devices and for MPIO devices with RDAC device drivers: • Harvested IBM.AgFileSystem resources can be automated. IBM.AgFileSystem resources are harvested if they are of type *ext2*, *ext3*, or *reiserfs* and reside on storage entities that are harvested themselves (storage entities of classes IBM.LogicalVolume, IBM.Partition, IBM.VolumeGroup, IBM.Disk). • User-defined IBM.AgFileSystem resources can be automated (for example, network file systems). • SCSI-2 reservation is supported for disks harvested from the RDAC driver. • Linux RAID (/dev/*md* devices) is supported. **Limitations:** • File systems created on *md* devices without using LVM will not be harvested or automated. • *md* devices themselves will not be harvested as IBM.Disk resources unless **pvcreate** has been issued for the /dev/*md* device. • SCSI-2 reservation is not supported for non-RDAC drivers nor for *md* devices themselves. • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster. • EVMS is not supported, which includes any volume groups or logical volumes created or managed by EVMS.

Support for single-path and multipath storage devices on other IBM System Storage disk systems

Table 82 on page 266 describes the storage resource manager support for single-path and multipath storage devices on other IBM System Storage disk systems.

Table 82. Storage resource manager support for single-path and multipath storage devices on other IBM System Storage disk systems

AIX	Linux on the System z hardware platform	Power Systems running on Linux Linux on the System x hardware platform
Full support is available for single-path I/O (SPIO) storage devices and for multipath I/O (MPIO) storage devices: • Harvested IBM.AgFileSystem resources can be automated. IBM.AgFileSystem resources are harvested if they are of type *GPFS*, *mmfs*, *jfs*, or *jfs2* and reside on storage entities that are harvested themselves (storage entities of classes IBM.LogicalVolume, IBM.VolumeGroup, IBM.Disk). • User-defined IBM.AgFileSystem resources can be automated (for example, network file systems). • SCSI-2 reservation is supported for SPIO devices and for MPIO devices using the Redundant Disk Array Controller (RDAC) driver, as well as the IBM SDDPCM driver with the single-path reserve policy for this SCSI-2 reservation. **Limitations:** • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster. • SCSI-2 reservation is not supported for SDD and SDDPCM MPIO drivers.	Only user-defined IBM.AgFileSystem resources can be automated. **Limitations:** • Resource harvesting is not supported. Even if harvesting is successful, the harvested resources cannot be automated. • SCSI reservation is not supported. • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster.	Full support is available for SPIO devices and limited support for MPIO devices for the IBM System Storage disk systems: • Harvested IBM.AgFileSystem resources can be automated. IBM.AgFileSystem resources are harvested if they are of type *GPFS*, *mmfs*, *ext2*, *ext3*, or *reiserfs* and reside on storage entities that are harvested themselves (storage entities of classes IBM.LogicalVolume, IBM.Partition, IBM.VolumeGroup, IBM.Disk). • User-defined IBM.AgFileSystem resources can be automated (for example, network file systems). • Any disk supported by Linux RAID (/dev/*md* devices) is supported. **Limitations:** • SCSI-2 reservation is supported for the RDAC driver only. • File systems created on *md* devices without using LVM will not be harvested or automated. • *md* devices themselves will not be harvested as IBM.Disk resources unless **pvcreate** has been issued for the /dev/*md* device. • EVMS is not supported, which includes any volume groups or logical volumes created or managed by EVMS. • RHEL 5, SLES 10, and SLES 11 support consists of harvesting storage entities of classes IBM.Disk, IBM.VolumeGroup, IBM.LogicalVolume, IBM.AgFileSystem, and IBM.Partition. File systems can be automated, subject to the previously stated limitations for *md* devices.

Support for single-path, non-DS4000 storage devices

Table 83 describes the storage resource manager support for single-path, non-DS4000 storage devices.

Table 83. Storage resource manager support for single-path, non-DS4000 storage devices

AIX	Linux on the System z hardware platform	Power Systems running on Linux Linux on the System x hardware platform
Full support is available for single-path storage devices: • Harvested IBM.AgFileSystem resources can be automated. IBM.AgFileSystem resources are harvested if they are of type *jfs* or *jfs2* and reside on storage entities that are harvested themselves (storage entities of classes IBM.LogicalVolume, IBM.VolumeGroup, IBM.Disk). • User-defined IBM.AgFileSystem resources can be automated (for example, network file systems). • SCSI-2 reservation is supported. **Limitations:** • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster.	Only user-defined IBM.AgFileSystem resources can be automated. **Limitations:** • Resource harvesting is not supported. Even if harvesting is successful, the harvested resources cannot be automated. • SCSI reservation is not supported. • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster.	Limited support is available: • Harvested IBM.AgFileSystem resources can be automated. IBM.AgFileSystem resources are harvested if they are of type *ext2*, *ext3*, or *reiserfs* and reside on storage entities that are harvested themselves (storage entities of classes IBM.LogicalVolume, IBM.Partition, IBM.VolumeGroup, IBM.Disk). • User-defined IBM.AgFileSystem resources can be automated (for example, network file systems). **Limitations:** • Support for SCSI reservation is limited. Perform a disk reserve operation to check whether SCSI reservation is available. • User-defined IBM.AgFileSystem resources can only be automated if the disk hosting the file system has the same device name on all nodes in the cluster.

Support for network file systems

Table 84 on page 267 describes the storage resource manager support for network file systems.

Table 84. Storage resource manager support for network file systems (NFS)

AIX	Linux on the System z hardware platform	Power Systems running on Linux Linux on the System x hardware platform
Network file systems are not harvested. To automate such file systems, you must use user-defined IBM.AgFileSystem resources.	Network file systems are not harvested. To automate such file systems, you must use user-defined IBM.AgFileSystem resources.	Network file systems are not harvested. To automate such file systems, you must use user-defined IBM.AgFileSystem resources.

Note: Network file systems of class IBM.AgFileSystem can only be automated and monitored successfully if the root user of the importing system has write access to the file system.

Configuring cluster security services

You can administer cluster security services for management domains and RSCT peer domains.

On AIX, Linux, and Solaris systems, RSCT's cluster security services provide the security infrastructure that enables RSCT components to authenticate and authorize the identity of other parties.

Authentication is the process of ensuring that a party is who it claims to be. Using cluster security services, such cluster applications as the configuration resource manager can check that other parties are genuine and not attempting to gain unwarranted access to the system.

Authorization is the process by which a cluster software component grants or denies resources based on certain criteria. Currently, the only RSCT component that implements authorization is RMC, which uses access control list (ACL) files in order to control user access to resource classes and their resource instances. In these ACL files, you can specify the permissions that a user must have in order to access particular resource classes and resources. The RMC component subsystem uses cluster security services to map the operating system user identifiers specified in the ACL file with network security identifiers to determine if the user has the correct permissions. This process of mapping operating system user identifiers to network security identifiers is called *native identity mapping*.

Administrative tasks related to cluster security services fall into the following broad task categories:

- Configuring the cluster security services library
- Configuring the host based authentication mechanisms
- Configuring the global and local authorization identity mappings

Related concepts:

"Cluster security services" on page 6
RSCT applications and components use cluster security services to perform authentication within both management and peer domains.

"An RSCT peer domain" on page 35
An *RSCT peer domain* is a cluster of nodes configured for high availability.

"RSCT security considerations on the Windows platform" on page 24
RSCT's cluster security services provide the security infrastructure that enables components of RSCT to authenticate and authorize the identity of other parties.

"Cluster security services authentication concepts" on page 6
Authentication is the process by which a cluster software component, using cluster security services, determines the identity of one of its peers, clients, or an RSCT subcomponent.

"Cluster security services authorization concepts" on page 10
Authorization is the process by which a cluster software component grants or denies access to resources based on certain criteria.

Related tasks:

"Creating an RSCT management domain" on page 207
A management domain is defined as a set of nodes whose resources can be managed and monitored from one of the nodes that is designated as the management control point (MCP). All other nodes are considered to be managed nodes. A managed node can be configured in multiple management domains

at the same time.

Related information:

"Administering an RSCT peer domain" on page 35
To achieve high availability, you can configure a cluster of nodes into an RSCT peer domain.

"Managing user access to resources using RMC ACL files" on page 126
RMC implements authorization using an *access control list* (ACL) file. Specifically, RMC uses the ACL file on a particular node to determine the permissions that a user must have in order to access resource classes and their resource instances.

Configuring the cluster security services library

This topic contains information about MPM configuration files. If you wish to disable any or all of the MPMs configured for the cluster security services, contact the IBM Support Center.

Note: IBM does not support a configuration where none of the supplied security mechanisms are active. Such a configuration effectively eliminates any security features of the cluster infrastructure.

Cluster security services provides a Mechanism Abstraction Layer (MAL) that converts the mechanism-independent instructions requested by an application into general tasks to be performed by any mechanism. A Mechanism Pluggable Module (MPM) is a component that converts generalized security services routines into the specific security mechanism functions. Table 85 shows the available MPMs provided by cluster security services and the security mechanism that they support:

Table 85. MPMs provided by the cluster security services

MPM mnemonic	MPM path name	Security mechanism
unix	**/usr/lib/unix.mpm**	Host based authentication (HBA)
hba2	**/usr/lib/hba2.mpm**	Enhanced host based authentication (HBA2)

When cluster security services is installed on a node, a default MPM configuration file is installed in **/usr/sbin/rsct/cfg/ctsec.cfg**. This is an ASCII text file that lists information for each MPM on the system. Figure 7 shows the contents of the **/usr/sbin/rsct/cfg/ctsec.cfg** configuration file.

```
#Prior Mnemonic         Code           Path                    Flags
#-------------------------------------------------------------------------
 1      unix             0x00001        /usr/lib/unix.mpm      i
 2      hba2             0x00002        /usr/lib/hba2.mpm      iz[unix]
```

Figure 7. Contents of the **/usr/sbin/rsct/cfg/ctsec.cfg** *configuration file*

The entries in the configuration file contain the mnemonic and path name of the MPM, an identification code number for the MPM, a priority value, and MPM instruction flags. The priority value indicates the preferred security mechanism for the node and specifies a priority order among multiple MPMs.

Mechanism priority order

MPM priority order can be altered by the cluster administrator, should a security mechanism become more or less preferred.

Before changing the default security mechanism priority order, ensure that the nodes within the cluster provide support for the mechanisms being considered for higher priority and ensure that access controls for the service nodes within the cluster grant access to clients that would be authenticated using the higher priority mechanism. If these issues are not addressed prior to changing the mechanism priority order, valid clients that were allowed access to resources may be denied access to resources after the new priority order takes effect. See "Cluster security services authorization concepts" on page 10 for more information on planning and establishing proper authorization and access control.

Note: None of the MPMs shipped with the cluster security services should be removed from the configuration file. Removing an entry should only be done if the MPM is known to be failing and only when instructed by the IBM Support Center.

Mechanism instruction flags

As a general rule, the instruction flags for the MPMs that are listed in the **ctsec.cfg** file must not be modified. The exception to this general rule is the alternate authorization mechanism flag for the enhanced host based authentication (HBA2) MPM.

Alternate authorization mechanism flag for the enhanced host based authentication (HBA2) MPM:

The HBA2 mechanism supports the use of an alternate authorization mechanism through the **z** instruction flag.

This feature is described in "Understanding alternate authorization mechanisms" on page 13. The default cluster security services library configuration enables this feature for the HBA2 MPM, specifying the HBA MPM as the mechanism to use for authorizing HBA2 authenticated clients.

Only the HBA MPM (known by the **unix** mnemonic) may be employed as an alternate authorization mechanism for the HBA2 MPM. No other MPM is supported in this capacity.

The HBA2 MPM does provide authorization features and, if desired, the use of the HBA MPM as the authorization mechanism can be removed. Before taking such action, cluster administrators must ensure that access controls throughout the cluster have been modified to allow HBA2 MPM clients to access resources. Once the alternate authorization mechanism is disabled, all clients authenticated using the HBA2 MPM will no longer be considered to be HBA MPM (**unix**) identities during authorization checks. For more information about authorization, see "Cluster security services authorization concepts" on page 10.

Modifying the cluster security services library configuration

If you have moved any of the **.mpm** files from their default location, or if you wish to change the priority order of the MPMs or alter any MPM instruction flags that are permitted to be changed, you will need to modify the configuration file.

Do this in one of the following ways:

- Manually copy and edit the file, as follows:
  1. Copy the **/usr/sbin/rsct/cfg/ctsec.cfg** file to **/var/ct/cfg/ctsec.cfg**.

     ```
     $ cp /usr/sbin/rsct/cfg/ctsec.cfg /var/ct/cfg/ctsec.cfg
     ```

     Do *not* modify the default configuration file in **/usr/sbin/rsct/cfg/**.
  2. Using an ASCII text editor, modify the new **ctsec.cfg** file in **/var/ct/cfg**.

     Do *not* modify the mnemonic or code value for any MPM. Also, do not modify the flag values for the HBA MPM (the MPM using the **unix** mnemonic) and do not remove the **i** flag from the HBA2 MPM.
- Use the **ctscfg** command.

  The **ctscfg** command will copy the configuration file to **/var/ct/cfg/ctsec.cfg**, if a copy does not already exist, and will add, delete, or update the MPM configuration information according to the flags you specify on the command.

  For complete syntax information on the **ctscfg** command, see its online man page. For detailed syntax information, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Configuring the host based authentication mechanisms

There are several administrative tasks related to the host based authentication mechanisms.

Table 86 describes the administrative tasks you may need or want to perform that are related to the host based authentication mechanisms.

Table 86. Administrative tasks related to the host based authentication mechanisms

Task	Describes how to...	Perform this task if...
"Configuring the ctcasd daemon on a node"	Modify a configuration file read by the Cluster Security Services daemon (**ctcasd**) upon startup.	You want to modify the operational parameters of the **ctcasd** daemon. You can configure such things as how many threads the daemon creates, the key generation methods it uses in preparing host public and private keys, and where the daemon looks for key files and the trusted host list.
"Configuring credential life span" on page 274	Set the credential life span for a security mechanism and, for the HBA2 mechanism, enable credential tracking.	You want the security mechanism to detect expired credentials and refuse authentication to applications that present such credentials. For the HBA2 mechanism, you also want to refuse authentication to applications that present previously authenticated credentials.
"Guarding against address and identify spoofing when transferring public keys" on page 276	Copy public keys between nodes to establish the security environment needed for a management domain or an RSCT peer domain.	You do not think your network security is sufficient to prevent address and identity spoofing. If you are confident in the security of your network, you do not need to perform this task; the keys will be copied automatically as part of your node configuration process.
"Changing a node's private/public key pair" on page 279	Modify a node's private and public keys.	A node's private key needs to be modified.

Configuring the ctcasd daemon on a node

When using host based authentication (HBA) or enhanced host based authentication (HBA2) as a security method, cluster security services uses the **ctcasd** daemon to provide and authenticate operating system identity based credentials.

The **ctcasd** daemon obtains its operational parameters from a configuration file (**ctcasd.cfg**). This configuration filesets the operational environment for the daemon, including:

- How many threads to create
- What key generation method to use in preparing host public and private keys
- Where the key files and trusted host list files reside on the node
- How long credentials should be considered valid after their creation
- Whether execution tracing should be enabled

When cluster security services are installed on a node, a default configuration file is installed in **/usr/sbin/rsct/cfg/ctcasd.cfg**. This is an ASCII text file that contains configurable parameters and their associated default values. ***This default configuration file should not be modified.*** If you wish to change the **ctcasd** configuration on a node to, for example, improve the performance of the daemon by altering the thread limits, use the following procedure:

1. Copy the **/usr/sbin/rsct/cfg/ctcasd.cfg** file to **/var/ct/cfg/ctcasd.cfg**.

   ```
   cp /usr/sbin/rsct/cfg/ctcasd.cfg /var/ct/cfg/ctcasd.cfg
   ```

2. Using an ASCII text editor, modify the new **ctcasd.cfg** file in **/var/ct/cfg**. The contents of the file will look similar to the following:

   ```
   TRACE= ON
   TRACEFILE= /var/ct/IW/log/ctsec/ctcasd/trace
   TRACELEVELS= _SEC:Info=1,_SEC:Errors=1
   TRACESIZE= 1003520
   RQUEUESIZE=
   MAXTHREADS=
   MINTHREADS=
   ```

```
THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
HBA_PRVKEYFILE=
HBA_PUBKEYFILE=
HBA_THLFILE=
HBA_KEYGEN_METHOD= rsa512
HBA_CRED_TIMETOLIVE=
HBA2_CRED_CTX_LIFETIME= -1
HBA2_CRED_TIMETOLIVE= 300
HBA2_NONCE_FILEMIN=
SERVICES=hba CAS
```

The keywords listed in this file will set the configurable parameters for the **ctcasd** daemon on this node. Table 87 describes the configurable parameters.

Table 87. Keyword descriptions for the ctcasd daemon configuration file

Keyword	Description
TRACE	Indicates whether or not tracing of the **ctcasd** daemon is enabled. Valid values are "on" and "off". When tracing is enabled, the TRACEFILE, TRACELEVELS, and TRACESIZE keywords specify the location, level, and size of the trace file generated. Setting the CT_TR_TRACE environment variable overrides any setting specified using the TRACE keyword in the **ctcasd.cfg** file. For more information about tracing the **ctcasd** daemon, see the *Troubleshooting RSCT* guide.
TRACEFILE	When tracing of the **ctcasd** daemon is enabled, this indicates the location of the trace file. If this value is not set, the default location is **/var/ct/IW/log/ctsec/ctcasd/trace**. The default directory **/var/ct/IW/log/ctsec/ctcasd** will be created automatically by the **ctcasd** daemon. However, if you use the TRACEFILE keyword to specify another location, you must ensure that the directory you specify exists. If it does not, the default location will be used instead, and an error will be logged in the trace. Setting the CT_TR_FILENAME environment variable overrides any setting specified using the TRACEFILE keyword in the **ctcasd.cfg** file.
TRACELEVELS	When tracing of the **ctcasd** daemon is enabled, the level of the trace. The _SEC category traces execution of the **ctcasd** daemon. Valid values are: **_SEC:Info=0** no tracing **_SEC:Info=1** trace minimum information messages **_SEC:Info=4** trace additional information messages **_SEC:Info=8** trace all information messages **_SEC:Errors=0** no tracing for errors **_SEC:Errors=1** trace all errors causing domain termination **_SEC:Errors=2** trace all call errors **_SEC:Errors=4** trace failed requests **_SEC:Errors=8** trace all errors

Table 87. Keyword descriptions for the ctcasd daemon configuration file (continued)

Keyword	Description
	The _SEU category traces processing within the **unix** host based authentication (HBA) MPM that can be invoked from the **ctcasd** daemon. Valid values are: **_SEU:Info=1** trace all informational messages **_SEU:Errors=1** trace all errors **_SEU:API=1** trace all entries and exits from HBA MPM interfaces **_SEU:API=8** trace entries, exits, and parameters from HBA MPM interfaces The _SEH category traces processing within the **hba2** enhanced host based authentication (HBA2) MPM that can be invoked from the **ctcasd** daemon. Valid values are: **_SEH:Info=1** trace basic informational messages **_SEH:Info=2** trace informational messages with more detail **_SEH:Info=8** trace all informational messages **_SEH:Errors=1** trace all errors **_SEH:API=1** trace all entries and exits from HBA2 MPM interfaces **_SEH:API=8** trace entries, exits, and parameters from HBA2 MPM interfaces The _SEI category traces processing within the native identity mapping functions that can be invoked from the **ctcasd** daemon. Valid values are: **_SEI:Error=1** trace all errors **_SEI:API=1** trace all entries and exits from the native identity mapping interfaces **_SEI:API=8** trace entries, exits, and parameters from the native identity mapping interfaces **_SEI:Mapping=1** reports the identity mapping rule employed to obtain a mapped identity **_SEI:Mapping=2** reports the identity obtained through the identity mapping procedure **_SEI:Mapping=8** combines the results of _SEI:Mapping levels 1 and 2 **_SEI:Milestone=1** indicates major processing checkpoints in the identity mapping process **_SEI:Milestone=8** traces details of major processing checkpoints in the identity mapping process **_SEI:Diag=1** traces diagnostic information for the IBM Support Center

Table 87. Keyword descriptions for the ctcasd daemon configuration file (continued)

Keyword	Description
	The trace settings can be combined by using a comma to separate each setting. For example: `TRACELEVELS= _SEC:Info=8,_SEC:Errors=8` If not specified, the default is _SEC:Info=1, _SEC:Errors=1. Setting the CT_TR_TRACE_LEVELS environment variable overrides any setting specified using the TRACELEVELS keyword in this file. For more information about tracing the **ctcasd** daemon, see the *Troubleshooting RSCT* guide.
TRACESIZE	When tracing of the **ctcasd** daemon is enabled, this indicates the size of the trace file. The minimum size is 4096, and the number specified will be rounded up to the nearest 4096 multiple. If not specified, the default trace-file size is 1003520. Setting the CT_TR_SIZE environment variable overrides any setting specified using the TRACESIZE keyword in the **ctcasd.cfg** file. For more information about tracing the **ctcasd** daemon, see the *Troubleshooting RSCT* guide.
RQUEUESIZE	Indicates the maximum length permitted for the daemon's internal run queue. If this value is not set, a default value of 64 is used.
MAXTHREADS	The limit to the number of working threads that the daemon may create and use at any given time (the “high water mark”). If this value is not set, a default value of 10 is used.
THREADSTACK	Sets the internal memory used by the daemon for thread stack space. The value is expressed in bytes. If no value is specified, the default system thread stack size is used. **You should not modify this value unless instructed to do so by the IBM Support Center.**
MINTHREADS	The number of idle threads that the daemon will retain if the daemon is awaiting further work (the “low water mark”). If this value is not, set, a default value of 4 is used.
HBA_USING_SSH_KEYS	Indicates if the daemon is making use of Secured Remote Shell keys. Acceptable values are true and false. If no value is provided, a default value of false is used. Secured Remote Shell keys are not supported in the current release.
HBA_PRVKEYFILE	Provides the full path name of the file that contains the local node's private key. The directories in the path must exist. If they do not exist, the **ctcasd** daemon will terminate. If this value is not set, the default location of **/var/ct/cfg/ct_has.qkf** is used.
HBA_PUBKEYFILE	Provides the full path name of the file that contains the local node's public key. The directories in the path must exist. If they do not exist, the **ctcasd** daemon will terminate. If this value is not set, the default location of **/var/ct/cfg/ct_has.pkf** is used.
HBA_THLFILE	Provides the full path name of the file that contains the local node's trusted host list. If any directory in the path does not exist, the **ctcasd** daemon will start without creating a trusted host list. If this value is not set, the default location of **/var/ct/cfg/ct_has.thl** is used.
HBA_KEYGEN_METHOD	Indicates the method to be used by **ctcasd** to generate the private and public keys of the local node if the files containing these keys do not exist. Acceptable values are those that can be provided as arguments to the **ctskeygen -m** command. If no value is provided for this attribute, the default value of **rsa1024** is used.
HBA_CRED_TIMETOLIVE	Sets the life span of host based authentication (HBA) credentials (credentials created and verified using the **unix** mnemonic MPM). The credential life span dictates the period of time after a credential is created that the HBA mechanism should consider the credential valid. Setting a credential life span enables the HBA mechanism to detect outdated credentials and refuse authentication to applications presenting such credentials. If no value is specified for this keyword (the default), then credentials will not be checked for expiration. For more information on using this keyword, see the “Configuring credential life span” on page 274.

Table 87. Keyword descriptions for the ctcasd daemon configuration file (continued)

Keyword	Description
HBA2_CRED_CTX_LIFETIME	Sets the expiration time for a security context that is established using the enhanced host based authentication (HBA2) mechanism. Once the security context is established, the context will remain valid for the length of time specified by this parameter. After this amount of time passes, the client and server applications will need to re-establish the security context. If no value is specified for this parameter, the HBA2 MPM will use a default value of 43 200 seconds (12 hours). The default **ctcasd.cfg** file sets this value to -1, indicating that security contexts established using the HBA2 MPM will not expire.
HBA2_CRED_TIMETOLIVE	Sets the life span of enhanced host based authentication (HBA2) credentials (credentials created and verified using the **hba2** mnemonic MPM). The credential life span dictates the period of time after a credential is created that the HBA2 mechanism should consider the credential valid. Setting a credential life span enables the HBA2 mechanism to detect outdated credentials and refuse authentication to applications presenting such credentials. If no value is specified for this keyword, then credential tracking is not performed and credentials will not be checked for expiration. The default **ctcasd.cfg** file sets this value to 300 seconds (5 minutes). For more information on using this keyword, see the "Configuring credential life span."
HBA2_NONCE_FILEMIN	Indicates the minimum number of credential identities retained by the enhanced host based authentication (HBA2) mechanism between executions of the **ctcasd** daemon. Whenever the HBA2 MPM authenticates a credential, the identity information for that credential is stored and used in subsequent authentication attempts to detect repeat uses of the same credential. The **ctcasd** daemon creates a file and reserves enough file system space so that the HBA2 MPM can store the minimum number of credential identities. When the **ctcasd** daemon starts, it reads the contents of this file into memory and uses it in subsequent authentication checks using the HBA2 MPM. This permits ctcasd and the HBA2 MPM to check for re-used credentials from prior executions of the daemon if the **ctcasd** daemon has been shut down. If no value is specified for this parameter, the **ctcasd** daemon uses a default value of 4096.
SERVICES	Lists the internal library services that the daemon supports. Do not modify this entry unless instructed to do so by the IBM Support Center.

3. Stop and restart the **ctcasd** daemon. Be aware that, while the daemon is offline, authentication will not be possible. To stop the daemon, issue the command:

   ```
   stopsrc -s ctcas
   ```

   To restart the daemon, issue the command:

   ```
   startsrc -s ctcas
   ```

Configuring credential life span

The **ctcasd.cfg** file's HBA_CRED_TIMETOLIVE and HBA2_CRED_TIMETOLIVE keywords set the credential life spans for their respective security mechanisms.

This is described in "Configuring the ctcasd daemon on a node" on page 270. The credential life span dictates the period of time after a credential is created that the security mechanism considers the credential valid. Setting a credential life span enables the host based authentication and enhanced host based authentication mechanisms to detect outdated credentials and refuse authentication to applications presenting such credentials. If no value is specified for these keywords, then credentials will not be checked for expiration.

The HBA2_CRED_TIMETOLIVE keyword has an additional impact on the enhanced host based authentication (HBA2) mechanism. When this keyword is set to a non-zero value, the **ctcasd** daemon tracks the credentials authenticated using the HBA2 MPM. Setting this keyword enables the **ctcasd**

daemon to identify credentials that may have been received previously and to refuse authentication to applications presenting previously authenticated credentials. Setting a value of `0` for this keyword disables this credential tracking.

Do not set a credential life span for the HBA or HBA2 MPMs on any node unless all nodes in your cluster have a common agreement on the current time of day. The time-of-day clocks on all systems within the operational cluster must be set to approximately the same time value. This requirement includes any Hardware Management Consoles (HMCs) that are contained within the operational cluster. Time zone differences between systems are permitted within the cluster, because the time-of-day clocks measure time in Universal Time Coordinated (UTC).

Sub-second time-of-day clock synchronization is not necessary for exploiting the credential life span capability of the HBA or HBA2 MPMs. The time-of-day clocks values need only be set within a reasonable tolerance of each other, typically within a few seconds. If your cluster makes use of a network time synchronization protocol such as NTP, the nodes of your cluster will already have a common agreement on the current time of day. If you are not using such a protocol, use the **date** command on the nodes of your cluster if their time-of-day clocks do not agree with each other.

The credential life span you specify using the HBA_CRED_TIMETOLIVE and HBA2_CRED_TIMETOLIVE keywords must allow for time-of-day clock differences between the systems in the operational cluster and the workload of these systems. If these factors are not considered when determining the credential life span, it is possible that credentials generated for applications on specific nodes will never be considered valid by specific service applications operating elsewhere within the same cluster.

Do the following to calculate an appropriate value for the credential life span:

1. Start with the desired credential life span value.
2. Add to this value the largest time-of-day clock difference between nodes of the operational cluster, including any Hardware Management Consoles (HMCs) in the cluster.
3. Add to this result the largest network latency time known for the nodes within the operational cluster.

Example: You have decided on a credential life span of 30 seconds. If the greatest time-of-day clock difference (in terms of Universal Time Coordinated) between two nodes is 23 seconds and the greatest network latency time between any set of systems on the cluster is estimated to be 8 seconds, then the credential life span should be set to 61 seconds.

Once you have decided on the appropriate credential life span, set the HBA_CRED_TIMETOLIVE and HBA2_CRED_TIMETOLIVE keywords on all systems within the operational cluster. The only exception to this rule is that the HBA_CRED_TIMETOLIVE keyword should not be set for any Hardware Management Consoles (HMCs) that exist within the cluster. While it is necessary that the time-of-day clocks on HMCs are set to approximately the same time value as all systems within the operational cluster, it is not necessary to set the HBA_CRED_TIMETOLIVE keyword on HMCs.

The default unit of measurement for the time interval specified using the HBA_CRED_TIMETOLIVE and HBA2_CRED_TIMETOLIVE keywords is seconds. The time value may be modified using the indicator "m" for minutes and "s" for seconds. Table 88 on page 276 shows some examples of valid specifications for the HBA_CRED_TIMETOLIVE keyword; the HBA2_CRED_TIMETOLIVE keyword behaves in the same manner.

Table 88. Examples of valid specifications for the HBA_CRED_TIMETOLIVE keyword

This specification...	Specifies a credential life span of...
HBA_CRED_TIMETOLIVE=	infinite (default)
HBA_CRED_TIMETOLIVE=0	infinite
HBA_CRED_TIMETOLIVE=90	90 seconds
HBA_CRED_TIMETOLIVE=90s	90 seconds
HBA_CRED_TIMETOLIVE=10m	10 minutes
HBA_CRED_TIMETOLIVE=10m 15	10 minutes and 15 seconds
HBA_CRED_TIMETOLIVE=10m 15s	10 minutes and 15 seconds

Guarding against address and identify spoofing when transferring public keys

When configuring a cluster of nodes, either as a management domain using CSM or RSCT commands, or as an RSCT peer domain using configuration resource manager commands, the necessary key exchanges between cluster nodes will, by default, be carried out automatically by CSM or by the appropriate RSCT resource manager.

These key exchanges between cluster nodes will be carried out as follows:

- In a management domain that is configured for CSM, the **updatenode** and **installnode** commands will, by default, copy the public key from each of the managed nodes to the management server, and will copy the management server's public key to each of the managed nodes. For more information on the **updatenode** and **installnode** commands, see *IBM Cluster Systems Management: Administration Guide*.
- In a management domain that is configured by xCAT, the **monadd rmcmon** and **moncfg rmcmon** commands will, by default, copy the public key from each of the managed nodes to the management server, and will copy the management server's public key to each of the managed nodes. For more information about the **monadd** and **moncfg** commands, refer to the xCAT documentation at **http://xcat.sourceforge.net**.
- In a management domain that is configured manually, as the **IBM.MngNode** and **IBM.MCP** resources are created, the public keys of the MCP and managed node are exchanged. See “Creating an RSCT management domain” on page 207.
- In an RSCT peer domain, the **preprpnode** command, when run on a particular node, will, by default, copy the public key from each of the remote nodes to the local node. Since the command will be run on each node in the domain, each node will have the public key information for all the other nodes in the domain. For information on the **preprpnode** command, see “Preparing the initial security environment on each node” on page 51.

Although the commands described above will automatically copy public keys to establish the necessary trust between nodes in the cluster, you must, before using the commands, consider whether the security of the network is sufficient to prevent address and identity spoofing. In a successful spoofing attack on a management domain, for example, a node might allow itself to be managed by the wrong "management server", or the wrong "managed node" might be invited into the network.

If you do not consider your network secure enough to avoid a possible spoofing attack, take the following precautions when configuring the cluster nodes:

- For an RSCT peer domain, manually transfer each node's public key to all other nodes in the RSCT peer domain and disable the **preprpnode** command's automatic key transferal. See “Manually transferring public keys” on page 277 for more information.
- For a management domain, verify the accuracy of the keys automatically transferred by the CSM or xCAT commands or by creation of **IBM.MngNode** and **IBM.MCP** resources. See “Verifying the accuracy of keys that are automatically transferred” on page 278 for more information.

Manually transferring public keys:

In an RSCT peer domain, you will need to copy each node's public key to all other nodes in the domain.

To manually transfer public keys:

1. Log on to the node being added to the RSCT peer domain.
2. Determine if the cluster has been set up to use fully qualified host names, short host names, or IP addresses for host based authentication. Make a note of the host name for this node in the corresponding format; this information will be required in step 5.
3. Issue the **ctsvhbal** command to obtain the list of available host based authentication mechanism identities for this system. The following output is displayed::

   ```
   ctsvhbal: The Host Based Authentication (HBA) mechanism identities for the
   local system are:

            Identity:  avenger.pok.ibm.com

            Identity:  9.117.10.4

   ctsvhbal: In order for remote authentication to be successful, at least one
   of the above identities for the local system must appear in the trusted host
   list on the remote node where a service application resides.  Ensure that at
   least one host name and one network address identity from the above list
   appears in the trusted host list on any remote systems that act as servers
   for applications executing on this local system.
   ```

4. Obtain the public key for this node by executing the following command:

   ```
   /usr/sbin/rsct/bin/ctskeygen -d > /tmp/hostname_pk.sh
   ```

   This command writes a text version of the local node's public key value to the file **/tmp/hostname_pk.sh**. The contents of this file will consist of two lines of output, similar to the following:

   ```
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
   e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
   ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
   c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
   03
   (generation method: rsa1024)
   ```

5. Edit the **/tmp/hostname_pk.sh** file, converting it to a shell script that issues the **ctsthl** command to insert this public key into a trusted host list file. Use the host name determined in step 2 as the argument to the **-n** option. Make sure that the field listed after the generation method field is used as the argument to the **-m** option of this command, and that the text version of the public key is used as the argument to the **-p** option. If the remote node will use a trusted host list file other than the default, list that file's name as an argument to the **-f** option; otherwise, omit the **-f** option. After editing the file, the contents of the file should resemble the following:

   ```
   /usr/sbin/rsct/bin/ctsthl -a -m rsa1024 -n avenger.pok.ibm.com -p
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
   e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
   ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
   c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
   03
   ```

6. Continue editing the **/tmp/hostname_pk.sh** file. Copy the instruction created in step 5 to a new line, and replace the host name argument of the **-n** option with a network address discovered in step 3. Repeat this process for each network address and host name discovered in step 3.

   Continuing with the previous example, the completed **/tmp/hostname_pk.sh** file would contain:

   ```
   /usr/sbin/rsct/bin/ctsthl -a -m rsa1024 -n avenger.pok.ibm.com -p
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
   e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
   ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
   c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
   ```

```
03
/usr/sbin/rsct/bin/ctsthl -a -m rsa1024 -n 199.100.100.4 -p
120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
03
/usr/sbin/rsct/bin/ctsthl -a -m rsa1024 -n 9.117.198.45 -p
120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
03
```

7. Transfer the **/tmp/hostname_pk.sh** shell script file to the remote node already within the cluster. This can be done via the **ftp** command, or by transferring this file to a diskette, transferring the diskette to the remote node, and reading the file off the diskette on the remote node.
8. Log on to the remote node.
9. Execute the **/tmp/hostname_pk.sh** shell script file on the node to add the new node's public key to the node's trusted host list:

   ```
   sh /tmp/hostname_pk.sh
   ```

10. Execute the **/usr/sbin/rsct/bin/ctsthl -l** command to verify that the key has been added to the trusted host list. An example host entry from the trusted host list as it appears in the **ctsthl** command output:

    ```
    --------------------
    Host name: avenger.pok.ibm.com
    Identifier Generation Method: rsa1024
    Identifier Value:
    120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
    e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
    ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
    c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
    03
    --------------------
    Host name: 199.100.100.4
    Identifier Generation Method: rsa1024
    Identifier Value:
    120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
    e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
    ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
    c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
    03
    --------------------
    Host name: 9.117.198.45
    Identifier Generation Method: rsa1024
    Identifier Value:
    120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
    e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
    ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
    c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
    03
    --------------------
    ```

Verifying the accuracy of keys that are automatically transferred:

When establishing a management domain, either through the use of CSM or xCAT commands or the creation of **IBM.MngNode** and **IBM.MCP** resources, public keys are automatically exchanged between the management server and managed nodes.

Keys are exchanged by copying:

- the public key from each of the managed nodes to the management server.
- the management server's public key to each of the managed nodes.

If you are concerned about potential address and identity spoofing in a management domain, you will need to verify that that correct keys are copied. To do this:

1. Log on to the node whose public key was copied.
2. Execute the following command on that node:

   ```
   /usr/sbin/rsct/bin/ctskeygen -d > /tmp/hostname_pk.sh
   ```

   This command writes a text version of the local node's public key value to the file **/tmp/hostname_pk.sh**. The contents of this file will consist of two lines of output, resembling the following:

   ```
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8e15a5dda5
   2499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407ccbfc98252072ee1c0
   381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3c34e23bb976eb55a386619b70c5
   dc9507796c9e2e8eb05cd33cebf7b2b27cf630103
   (generation method: rsa1024)
   ```
3. Log on to the remote node where the key was transferred.
4. Execute the **/usr/sbin/rsct/bin/ctsthl -l** command and verify that the correct key has been added to the trusted host list. The **ctsthl** command output should list entries for the host name and IP address(es) of the node. An example host entry from the trusted host list as it appears in the **ctsthl** command output:

   ```
   --------------------
   Host name: avenger.pok.ibm.com
   Identifier Generation Method: rsa1024
   Identifier Value:
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
   e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
   ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
   c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
   03
   --------------------
   Host name: 199.100.100.4
   Identifier Generation Method: rsa1024
   Identifier Value:
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
   e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
   ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
   c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
   03
   --------------------
   Host name: 9.117.198.45
   Identifier Generation Method: rsa1024
   Identifier Value:
   120400cc75f8e007a7a39414492329dcb5b390feacd2bbb81a7074c4edb696bcd8
   e15a5dda52499eb5b641e52dbceda2dccb8e8163f08070b5e3fc7e355319a84407
   ccbfc98252072ee1c0381bdb23fb686d10c324352329ab0f38a78b437b235dd3d3
   c34e23bb976eb55a386619b70c5dc9507796c9e2e8eb05cd33cebf7b2b27cf6301
   03
   --------------------
   ```

Changing a node's private/public key pair

In general, a node's private and public key pair are considered synonymous with a node's identity and are not expected to change over time.

However, if they do need to be changed, be aware that a node's private/public key pair should not be changed while a node is operational within the cluster. This is because it is difficult to synchronize a change in a node's public key on all the nodes that need the revised key. The unsynchronized keys will lead to failure in the applications that use cluster security services.

If a node's private key becomes compromised, it is impossible to tell for how long a private key may have been public knowledge or have been compromised. Once it is learned that such an incident has occurred, the system administrator must assume that unwarranted access has been granted to critical

system information for an unknown amount of time, and the worst must be feared in this case. Such an incident can only be corrected by a disassembly of the cluster, a reinstall of all cluster nodes, and a reformation of the cluster.

Configuring the global and local authorization identity mappings

The identity mapping service uses information stored in the identity mapping files **ctsec_map.global** (which contains the common, cluster-wide, identity mappings) and **ctsec_map.local** (which contains identity mappings specific to the local node only).

This is described in "Cluster security services authorization concepts" on page 10. These are ASCII-formatted files that you can modify using a text editor, thus enabling you to configure the global and local identity mappings.

Table 89 describes what you need to do to configure global and local authorization identity mappings.

Table 89. Configuring global and local authorization identity mappings

If you want to configure...	Then do this...
Global identity mappings	Add entries to the **/var/ct/cfg/ctsec_map.global** file on every node in the cluster. Entries ***must not*** be added to the default **/usr/sbin/rsct/cfg/ctsec_map.global** file. If the file **/var/ct/cfg/ctsec_map.global** file does not exist on a node, copy the default **/usr/sbin/rsct/cfg/ctsec_map.global** file to the **/var/ct/cfg** directory, and then add the new entries to the **/var/ct/cfg/ctsec_map.global** file. It is important that you do not remove any entries from the **/var/ct/cfg/ctsec_map.global** file that exist in the default file. It is also important that the **/var/ct/cfg/ctsec_map.global** files on all nodes within the cluster be identical.
Local identity mappings	Create the **/var/ct/cfg/ctsec_map.local** file on the local node and add entries to it. Be aware that RSCT does not provide a default **ctsec_map.local** file; you must create it yourself.

When creating **/var/ct/cfg/ctsec_map.global** and **/var/ct/cfg/ctsec_map.local** files, make sure the files can be read by any system user, but that they can be modified only by the root user (or other restrictive user identity not granted to normal system users). By default, these files reside in locally-mounted file systems. While it is possible to mount the **/var/ct/cfg** directory on a networked file system, we discourage this. If the **/var/ct/cfg/ctsec_map.local** file were to reside in a networked file system, any node with access to that networked directory would assume that these definitions were specific to that node alone when in reality they would be shared.

Each line in the **ctsec_map.global** and **ctsec_map.local** files is an entry. Each entry is used to either associate a security network identifier with a local operating system identifier, or else is used to expressly state that no association is allowed for a particular security network identifier. Lines that start with a pound sign (#) are considered comments and are ignored by the identity mapping service. Blank lines are also ignored by the identity mapping service, so you may include them to improve the readability of the files.

Each entry takes the form:

```
mechanism_mnemonic:identity_mapping
```

Where:

mechanism_mnemonic
: is the mnemonic used to represent the security mechanism in the MPM configuration file (as described in "Configuring the cluster security services library" on page 268). Currently, the supported security mechanisms are:

  Mnemonic
  : **Security mechanism**

  unix
  : Host based authentication (HBA) mechanism

hba2 Enhanced host based authentication (HBA2) mechanism

identity_mapping
is either an explicit mapping or a mapping rule. An *explicit mapping* maps a specified network security identifier with a specified local user identifier. A *mapping rule* uses pattern matching and MPM reserved words to determine which security network identifier(s) and local user identifier(s) are mapped.

Both the explicit mappings and the mapping rules can be either affirmative or negative. The *affirmative mappings* are the implied type of mapping; they associate network security identifiers with local user identifiers. The *negative mappings* explicitly state that no association is allowed for one or more network security identifiers.

The exact format of the identity mapping depends on the security mechanism. The MPM that supports the security mechanism can support its own mapping entry format, special characters, and reserved words. For more information on the format of identity mapping entries for the HBA and HBA2 mechanisms, see "Configuring the host based authentication (HBA) mechanism mappings" and "Configuring the enhanced host based authentication (HBA2) mechanism mappings" on page 284.

Since the native identity mapping information is spread out across two files (**ctsec_map.global** and **ctsec_map.local**), it is important to understand how the identity mapping service uses both these files. The identity mapping service parses the **ctsec_map.global** and **ctsec_map.local** files as follows:

1. If the **/var/ct/cfg/ctsec_map.local** file exists, the identity mapping service checks for associations in this file.
2. If the **/var/ct/cfg/ctsec_map.global** file exists, the identity mapping service checks for associations in this file.
3. If the **/var/ct/cfg/ctsec_map.global** file does not exist, then the identity mapping service checks for associations in the default file **/usr/sbin/rsct/cfg/ctsec_map.global**.

The identity mapping is performed on a first-match basis. In other words, the first mapping entry for a security network identity (regardless of whether it is an explicit mapping or a mapping rule) is the one applied. For this reason, the order of entries in the mapping file is important; you should place the most restrictive entries before the more relaxed ones. In particular, place entries containing explicit mappings before entries containing mapping rules. Also be aware that, if both the **ctsec_map.global** and **ctsec_map.local** files grant different associations to the same security network identifier, the identity mapping service will use the association stated by the entry in the **ctsec_map.local** file.

Since a single security network identifier may have multiple mapping entries in the mapping file(s), it may not be obvious which mapping is being obtained by the identity mapping service. If authorization is not working as expected, you may want to verify the identity mapping. You can do this using the **ctsidmck** command. The **ctsidmck** command verifies the mapping that would be obtained by the identity mapping service for a specified network identifier.

Example: To obtain the mapped identity for the host based authentication mechanism's security network identifier *zathras@greatmachine.epsilon3.org*, you would enter the following at the command prompt:

```
ctsidmck -m unix zathras@greatmachine.epsilon3.org
```

For complete information on the **ctsec_map.global** and **ctsec_map.local** files, and on the **ctsidmck** command, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Configuring the host based authentication (HBA) mechanism mappings

To indicate that an entry in the **ctsec_map.global** or **ctsec_map.local** file refers to the HBA mechanism, you must begin the entry with the `unix:` mnemonic.

For example. the following entry associates the HBA network identifier *jbrady@epsilon3.ibm.com* to the local user identifier *jbrady*:

```
unix:jbrady@epsilon3.ibm.com=jbrady
```

The preceding example is an affirmative explicit mapping—a specified security network identifier is associated with a specified local user identifier.

To create a negative mapping—a mapping that explicitly states that no association is allowed for a particular security network identifier—use the **!** reserved character .

Example: The following entry denies any local user identity association for the HBA network identifier *jbrady@epsilon3.ibm.com*:

```
unix:!jbrady@epsilon3.ibm.com
```

Usually, the HBA mechanism mappings will use host names as in the preceding examples. However, you can also create mappings using IP address character strings (in IPv4 or IPv6 format).

Example: The following entry is an affirmative explicit mapping using an IP address (instead of a host name):

```
unix:jbrady@9.117.10.14=jbrady
```

Example: The following entry is a negative explicit mapping using an IP address:

```
unix:!jbrady@9.117.10.14
```

However, be aware that IP-address authorization is only possible in an RSCT peer domain in which all nodes are using version 2.3.1.0 of RSCT or later. RSCT versions prior to 2.3.1.0 do not support IP-address authorization. In addition, IP-address authorization is not supported in a CSM management domain. In some cluster configurations, authorization might be based on IP addresses for some nodes and on host names for others. In these cases, you might want to create multiple mapping entries for the same host—one using the IP address and one using the host name.

Example: The following entries map the same node by host name and by IP address.

```
unix:jbrady@epsilon2.ibm.com=jbrady
unix:jbrady@9.117.10.14=jbrady
```

Using wildcard characters in host based authentication (HBA) mappings:

You can use the * wildcard character to match multiple user names or host names in the security network identifier.

If an entry uses the * wildcard character to match all user names in the security network identifier, it can also use the * wildcard character as the local user identifier. If it does, then the identity mapping service will associate each security network identifier to the local user identifier that matches the user name from the security network identifier. This is the only situation when you can use the * wildcard character in the local user identifier specification. You also cannot use the * wildcard character in place of the security mechanism mnemonic; you must explicitly specify the mnemonic.

Table 90 on page 283 shows several examples of how an entry can use the * wildcard character when specifying the user name portion of the security network identifier.

Table 90. Examples of using the wildcard character to match multiple user names in the security network identifier

This mapping entry...	Does this...
`unix:*@epsilon3.ibm.com=jbrady`	Associates any host based authentication mechanism network identifier from the host *epsilon3.ibm.com* with the local user identifier *jbrady*.
`unix:!*@epsilon3.ibm.com`	Explicitly states that no association is allowed for any HBA mechanism network identifier from the host *epsilon3.ibm.com*.
`unix:j*@epsilon3.ibm.com=jbrady`	Associates any HBA mechanism network identifier starting with the letter "j" from the host *epsilon3.ibm.com* with the local user identifier *jbrady*.

The information in the preceding table also applies when you identify a host using its IP address. For example, the entry:

```
unix:*@9.117.10.14=jbrady
```

associates any HBA identifier from the host `9.117.10.14` with the local user identifier *jbrady*.

You can only use the * wildcard character once within the user name specification. For example the entry:

```
unix:*athra*@epsilon3.ibm.com=zathras
```

is invalid since the entry repeats the * wildcard character between the token separators **:** and **@**.

Table 91 shows several examples of how an entry can use the * wildcard character when specifying the host identification portion of the security network identifier.

Table 91. Examples of using the wildcard character to match multiple host names in the security network identifier

This mapping entry...	Does this...
`unix:jbrady@*=jbrady`	Associates any HBA mechanism network identifier (host name or IP address) that contains the user name *jbrady* (regardless of the host) to the local user identifier *jbrady*.
`unix:!jbrady@*`	Explicitly states that no association is allowed for any HBA mechanism network identifier (host name or IP address) that contains the user name *jbrady* (regardless of the host).
`unix:zathras@*.ibm.com=zathras`	Associates any HBA mechanism network identifier that contains the user name *zathras* and a host name ending with the *ibm.com*® network domain to the local user identifier *zathras*.

When the * wildcard character replaces the entire host identification specification (for example, `jbrady@*`), it represents any host name or IP address.

You can only use the * wildcard character once within the host identification specification. For example the entry:

```
unix:zathras@*.ibm.*=zathras
```

is invalid since the entry repeats the * wildcard character between the token separators **@** and **=**.

The most powerful use of the * wildcard character is to associate each security network identifier with the local user identifier that matches the user name from the security network identifier. Table 92 on page 284 shows several examples of this.

Table 92. Examples of using the wildcard character to associate each security identifier with the local user identifier that matches the user name

This mapping entry...	Does this...
`unix:*@epsilon3.ibm.com=*`	Associates any HBA mechanism network identifier from the host *epsilon3.ibm.com* to the local user identifier that matches the user name from the security network identifier. For example, *zathras@epsilon3.ibm.com* will be associated with the local user identifier *zathras*, and *jbrady@epsilon3.ibm.com* will be associated with the local user identifier *jbrady*.
`unix:*@*=*`	Associates any HBA mechanism network identifier (host name or IP address) from any host to the local user identifier that matches the user name from the security network identifier. For example, *zathras@epsilon3.ibm.com* will be associated with the local user identifier *zathras*, and *jbrady@zaphod.ibm.com* will be associated with the local user identifier *jbrady*.

Using MPM-defined reserved words in host based authentication (HBA) mechanism mappings:

In addition to the wildcard character, there are three MPM-defined reserved words you can use when configuring the host based authentication mechanism.

The reserved words are: **<iw>**, **<cluster>**, and **<any_cluster>**.

The **<iw>** reserved word refers to the local node.

Example: The following entry associates the security network identifier *tardis* on the local node with the local user *root*:

```
unix:tardis@<iw>=root
```

The **<cluster>** reserved word refers to any host in the currently active cluster.

Example: The following entry associates any security network identifier that contains the user name *tardis* and originates from any host in the currently active cluster with the local user *root*:

```
unix:tardis@<cluster>=root
```

Therefore, if the hosts *anglashok.ibm.com* and *mimbar.ibm.com* are active in the cluster, then the identity mapping service will associate *tardis@anglashok.ibm.com* and *tardis@mimbar.ibm.com* with the local user *root*.

The **<any_cluster>** reserved word refers to any host within any cluster in which the local node is currently defined.

Example: The following entry associates any security network identifier that contains the user name *tardis* and originates from any host in any cluster in which the local node is defined with the local user *root*:

```
unix:tardis@<any_cluster>=root
```

Therefore, if the hosts *anglashok.ibm.com* and *mimbar.ibm.com* are defined within any cluster in which the local node is defined, then the identity mapping service will associate *tardis@anglashok.ibm.com* and *tardis@mimbar.ibm.com* with the local user *root*.

Configuring the enhanced host based authentication (HBA2) mechanism mappings

Enhanced host based authentication (HBA2) network identities are mapped to native user identities in the same manner as host based authentication (HBA) identities.

This is described in "Configuring the host based authentication (HBA) mechanism mappings" on page 281.

Native identity mapping for HBA2 network identities follows the same formats and rules as those described earlier for HBA network identities. HBA2 network identities also support the same negative mappings, wildcard substitution rules, and reserved words.

Restriction: Configuring the HBA2 security mechanism for use in a peer domain is currently not supported.

To indicate that an entry in the **ctsec_map.global** or **ctsec_map.local** file refers to the enhanced host based authentication mechanism, you must begin the entry with the `hba2:` mnemonic.

Example: The following entry is an affirmative explicit mapping that associates the HBA2 network identifier *jbrady@epsilon3.ibm.com* to the local user identifier *jbrady*.

```
hba2:jbrady@epsilon3.ibm.com=jbrady
```

Example: The following entry illustrates a negative mapping for an HBA2 network identity.

```
hba2:!jbrady@epsilon3.ibm.com
```

The HBA2 MPM also supports the use of IP addresses in authentication, as illustrated in the following examples.

Example: The following entry is an affirmative explicit mapping using an IP address.

```
hba2:jbrady@9.117.10.14=jbrady
```

Example: The following entry is a negative mapping using an IP address.

```
hba2:!jbrady@9.117.10.14
```

As with the HBA mechanism, the HBA2 mechanism can authenticate using host names from some cluster nodes and IP addresses from other cluster nodes. In these cases, it is best to create multiple mapping entries for the same host—one that uses the host name of the remote cluster node and one for each IP address supported by the remote cluster node.

Example: The following entries map the same node by host name and by IP address.

```
hba2:jbrady@epsilon2.ibm.com=jbrady
hba2:jbrady@9.117.10.14=jbrady
hba2:jbrady@9.118.102.49=jbrady
```

Using wildcard characters in enhanced host based authentication (HBA2) mappings:

HBA2 network identities follow the same wildcard use and substitution semantics as those used for HBA network identities.

See “Using wildcard characters in host based authentication (HBA) mappings” on page 282 for more information about using wildcard characters.

Using MPM-defined reserved words in enhanced host based authentication (HBA2) mechanism mappings:

HBA2 network identities can employ the same reserved words as those defined for the HBA mechanism.

See “Using MPM-defined reserved words in host based authentication (HBA) mechanism mappings” on page 284 for more information about using the reserved words.

Impacts of alternate authorization mechanisms on identity mapping

An alternate authorization mechanism instructs the mechanism abstraction layer (MAL) to treat a typed network identity as if it were authenticated by a *different* MPM than the MPM that authenticated the identity whenever the MAL performs native identity mapping and authorization.

This is discussed in "Understanding alternate authorization mechanisms" on page 13.

When an alternate authorization mechanism is configured for a specific MPM, the identity mapping procedure treats the network identity as if it were authenticated by the MPM specified as the alternate authorization mechanism. Identity mapping will proceed, but the rules for mapping identities for the alternate authorization mechanism will be applied to the network identity, *not* the rules used by the MPM that actually authenticated that identity.

An alternate authorization mechanism is specified within the cluster security services' configuration file **ctsec.cfg** using the **z** flag, followed by the mnemonic of the alternate authorization mechanism to be used, enclosed in brackets, as shown in Figure 8.

```
#Prior Mnemonic Code     Path               Flags
#-------------------------------------------------
 1     unix     0x00001  /usr/lib/unix.mpm  i
 2     hba2     0x00002  /usr/lib/hba2.mpm  iz[unix]
```

Figure 8. Cluster security services configuration file containing an alternate authorization mechanism

In Figure 8, the MPM with the mnemonic of **unix** is specified as the alternate authorization mechanism for the MPM with the mnemonic of **hba2**. Any network identities authenticated by the **hba2** MPM will be handled by the identity mapping process as if those identities were authenticated by the **unix** MPM.

Example: Based on the configuration shown in Figure 8, consider the case where a client is authenticated by the **hba2** MPM as *ranger1@ialliance.gov*. The identity mapping files contain the following rules:

```
unix:ranger1@ialliance.gov=valen
hba2:ranger1@ialliance.gov=sinclair
```

Because the **unix** MPM has been specified as the alternate authorization mechanism, the identity mapping process does not use the rule specified for the **hba2** identity in this case. Instead, the rule specified for the **unix** MPM is employed, which results in the mapping of this network identity to the native user identity of *valen*.

Note: The cluster security services do not support an alternate authorization mechanism for the host based authentication (HBA) mechanism, known by the **unix** mnemonic. Do *not* alter the configuration to specify an alternate authorization mechanism for the **unix** MPM.

Before specifying an alternate authorization mechanism for an MPM, it is important to understand how any native identity mapping rules may apply to the network identity that the authenticating MPM will provide. Some mechanisms may provide typed network identities for clients that contain special characters or keywords that the alternate authorization mechanism might interpret as wildcard characters or reserved words.

Note: The default cluster security services configuration installed by RSCT specifies an alternate authorization mechanism for the enhanced host based authentication (HBA2) mechanism. This alternate authorization mechanism is the HBA mechanism, known by its **unix** mnemonic. The same identity mapping rules, wildcard substitutions, and reserved words apply to both mechanisms. This reduces the concern of whether inappropriate rules are applied to the HBA2 network identities by the HBA identity mapping process.

The Topology Services subsystem

In an RSCT peer domain, the configuration resource manager uses the Topology Services subsystem to monitor the liveness of the adapters and networks included in communication groups.

The communication groups are created automatically when you bring the cluster (RSCT peer domain) online or when you explicitly create a group using the **mkcomg** command.

Related concepts:

"An RSCT peer domain" on page 35
An *RSCT peer domain* is a cluster of nodes configured for high availability.

"Topology Services subsystem" on page 14
The *Topology Services subsystem* (or *Topology Services*) provides other RSCT applications and subsystems within an RSCT peer domain with network adapter status, node connectivity information, and a reliable messaging service.

Related tasks:

"Adding nodes to the peer domain" on page 62
Use the **addrpnode** command to add one or more nodes to an RSCT peer domain definition, passing it the IP address or DNS name of the node you want to add.

Introducing Topology Services

Topology Services is a distributed subsystem of the IBM Reliable Scalable Cluster Technology (RSCT) software.

The RSCT software provides a set of services that support high availability on your system. Another service in the RSCT software is the Group Services distributed subsystem described in "The Group Services subsystem" on page 308. Both of these distributed subsystems operate within a domain. A domain is a set of machines upon which the RSCT components execute and, exclusively of other machines, provide their services.

Topology Services provides other high availability subsystems with network adapter status, node connectivity information, and a reliable messaging service. The adapter status and node connectivity information is provided to the Group Services subsystem upon request, Group Services then makes it available to its client subsystems. The Reliable Messaging Service, which takes advantage of node connectivity information to reliably deliver a message to a destination node, is available to the other high availability subsystems.

This adapter status and node connectivity information is discovered by an instance of the subsystem on one node, participating in concert with instances of the subsystem on other nodes, to form a ring of cooperating subsystem instances. This ring is known as a heartbeat ring, because each node sends a heartbeat message to one of its neighbors and expects to receive a heartbeat from its other neighbor. Actually each subsystem instance can form multiple rings, one for each network it is monitoring. This system of heartbeat messages enables each member to monitor one of its neighbors and to report to the heartbeat ring leader, called the Group Leader, if it stops responding. The Group Leader, in turn, forms a new heartbeat ring based on such reports and requests for new adapters to join the membership. Every time a new group is formed, it lists which adapters are present and which adapters are absent, making up the adapter status notification that is sent to Group Services.

In addition to the heartbeat messages, connectivity messages are sent around all rings. Connectivity messages for each ring will forward its messages to other rings, so that all nodes can construct a connectivity graph. It is this graph that determines node connectivity and defines a route that Reliable Messaging would use to send a message between any pair of nodes that have connectivity.

For more detail on maintaining the heartbeat ring and determining node connectivity, see "Topology Services components."

Topology Services components

The Topology Services subsystem consists of the following components:

Topology Services Daemon
: The central component of the Topology Services subsystem.

Pluggable Network Interface Module (NIM)
Program invoked by the Topology Services daemon to communicate with each local adapter.

Port numbers
TCP/IP port numbers that the Topology Services subsystem uses for daemon-to-daemon communications. The Topology Services subsystem also uses UNIX domain sockets for server-to-client and server-to-NIM communication.

Control command
A command that is used to add, start, stop, and delete the Topology Services subsystem, which operates under the SRC subsystem.

Startup command
A command that is used to obtain the configuration from the RSCT peer domain data server and start the Topology Services Daemon. This command is invoked by the SRC subsystem.

Tuning command
A command that is used to change the Topology Services tunable parameters at run-time.

Files and directories
Various files and directories that are used by the Topology Services subsystem to maintain run-time data.

The Topology Services daemon (hatsd)

The Topology Services daemon is the executable file **/usr/sbin/rsct/bin/hatsd**. This daemon runs on each node in an RSCT peer domain. The RSCT peer domain is the operational domain of the Topology Services subsystem.

When each daemon starts, it first reads its configuration from a file that is set up by the startup command (**cthats**). This file is called the *machines list file,* because it has all of the machines (nodes) listed that are part of the configuration and the IP addresses for each adapter for each of the nodes in that configuration. From this file, the daemon knows the IP address and node number of all the potential heartbeat ring members.

The directive of the Topology Services subsystem is to form as large a heartbeat ring as possible. To form this ring, the daemon on one node must alert those on the other nodes of its presence by sending a *proclaim message.* According to a hierarchy that is defined by the Topology Services component, daemons can send a proclaim message only to IP addresses that are lower than its own and can accept a proclaim message only from an IP address higher than its own. Also, a daemon only proclaims if it is the leader of a ring. When a daemon first starts up, it builds a heartbeat ring for every local adapter, containing only that local adapter. This is called a *singleton group.* This daemon is the group leader in each one of these singleton groups.

To manage the changes in these groups, the Topology Services subsystem defines the following roles for each group:

group leader
The daemon on the node with the local adapter that has the highest IP address in the group. The group leader proclaims, handles request for joins, handles death notifications, coordinates group membership changes, and sends connectivity information.

group leader successor
The daemon on the node with the local adapter that has the second highest IP address in the group. This daemon can detect the death of the group leader and has the authority to become the group leader of the group if that happens.

mayor A daemon on a node with a local adapter present in this group that has been picked by the group leader to broadcast a message to all the adapters in the group. When a daemon receives a message to broadcast, it is a mayor.

generic

The daemon on any node with a local adapter in the heartbeat ring. The role of the generic daemon is to monitor the heartbeat of the upstream neighbor and inform the group leader if the maximum allowed number of heartbeats have been missed.

Each one of these roles is dynamic, which means that every time a new heartbeat ring is formed, the roles of each member are evaluated and assigned.

In summary, group leaders send and receive proclaim messages. If the proclaim is from a group leader with a higher IP address, then the group leader with the lower address replies with a join request. The higher address group leader forms a new group with all members from both groups. All members monitor their upstream neighbor for heartbeats. If a sufficient number of heartbeats are missed, a message is sent to the group leader and the unresponsive adapter will be dropped from the group. Whenever there is a membership change, Group Services is notified if it asked to be.

The group leader also accumulates node connectivity information, constructs a connectivity graph, and routes connections from its node to every other node in the RSCT peer domain. The group connectivity information is sent to all nodes so that they can update their graphs and also compute routes from their node to any other node. It is this traversal of the graph on each node that determines which node membership notification is provided to each node. Nodes to which there is no route are considered unreachable and are marked as "down". Whenever the graph changes, routes are recalculated, and a list of nodes that have connectivity is generated and made available to Group Services.

Pluggable NIMs

The Topology Services subsystem's pluggable network information and monitoring services (NIMs) are processes that the Topology Services daemon starts to monitor each local adapter.

The NIM is responsible for:

1. Sending messages to a peer daemon upon request from the local daemon.
2. Receiving messages from a peer daemon and forwarding it to the local daemon.
3. Periodically sending heartbeat messages to a destination adapter.
4. Monitoring heartbeats that come from a specified source and notifying the local daemon if any heartbeats are missing.
5. Informing the local daemon if the local adapter goes up or down.

When a network adapter fails or has a problem in one node, incoming heartbeats can be lost. To distinguish a local adapter failure from a remote adapter (or network) failure, the NIM calls a network monitor library of functions to determine the local adapter's health. This "netmon" logic examines the adapter statistics to determine whether it is able to receive data packets from the network. If the network is not already naturally active, the NIM tries to stimulate new traffic by using Internet Control Message Protocol (ICMP) echo requests, commonly known as sending "pings." If the adapter is still not receiving any new traffic even after the NIM exercised several rounds of ping attempts, it is considered to be down.

After the NIM notified the daemon that the local adapter is down, the daemon notifies Group Services that all adapters in the group are down. No other node in the cluster can be reachable by using that path if the local interface is down.

After an adapter that was down recovers, the NIM detects that the adapter is working again through the same netmon logic, and the daemon then forms a singleton group with it. This process then allows the daemon to form a larger group with the other adapters in the network, and it sends an "adapter up" notification for the local adapter to the Group Services subsystem.

Port numbers and sockets

The Topology Services subsystem uses the following types of communication:

- UDP port numbers for intracluster communication, that is, communication between Topology Services daemons within the RSCT peer domain
- UNIX domain sockets for communication between:
  1. The Topology Services clients and the local Topology Services daemon
  2. The local Topology Services daemon and the NIMs

Intracluster port numbers:

For communication between Topology Services daemons within the RSCT peer domain, the Topology Services subsystem uses a single UDP port number.

This port number is provided by the configuration resource manager during cluster creation. You supply the UDP port number using the **-t** flag on the **mkrpdomain** command (as described in "Creating a peer domain definition" on page 52).

The Topology Services port number is stored in the cluster data so that, when the Topology Services subsystem is configured on each node, the port number is retrieved from the cluster data. This ensures that the same port number is used by all Topology Services daemons in the RSCT peer domain.

This intracluster port number is also set in the **/etc/services** file, using the service name **cthats**. The **/etc/services** file is updated on all nodes in the RSCT peer domain.

UNIX domain sockets:

The UNIX domain sockets used for communication are connection-oriented sockets.

For the communication between between the Topology Services clients and the local Topology Services daemon, the socket name is /var/ct/*cluster_name*/soc/cthats/server_socket, where *cluster_name* is the name of the RSCT peer domain. For the communication between the local Topology Services daemon and the NIMs, the socket name is /var/ct/*cluster_name*/soc/cthats/*NIM_name.process_id*, where *cluster_name* is the name of the cluster (RSCT peer domain), *NIM_name* is the name of the NIM, and *process_id* is the PID.

The control command (cthatsctrl)

The Topology Services control command is the executable file **/usr/sbin/rsct/bin/cthatsctrl**. In the normal operation of a cluster, this command should never need to be invoked manually.

In an RSCT peer domain, the configuration resource manager controls the Topology Services subsystem, so using the **cthatsctrl** command directly could yield undesirable results. In an RSCT peer domain, use this command only if an IBM service representative instructs you to do so.

The purpose of the **cthatsctrl** command is to add the Topology Services subsystem to the operating software configuration of the cluster. You can also use this command to remove the subsystem from the cluster, start the subsystem, stop the subsystem, and build the configuration file for the subsystem.

The startup command (cthats)

The Topology Services startup command **cthats** is the executable file **/usr/sbin/rsct/bin/cthats**.

The **cthats** command obtains the necessary configuration information from the cluster data server and prepares the environment for the Topology Services daemon. Under normal operating conditions, the Topology Services startup command runs without user initiation. When you run the **mkcomg** command or the **startrpdomain** command, the configuration resource manager starts the Topology Services subsystem automatically. If a problem occurs, you might need to run the **cthatsctrl** command to operate the Topology Services subsystem.

For more information about:

- The **mkcomg** command, see "Creating a communication group" on page 77
- The **startrpdomain** command, see "Bringing the peer domain online" on page 57

The tuning command (cthatstune)

The Topology Services tuning command **cthatstune** is the executable file **/usr/sbin/rsct/bin/cthatstune**.

The purpose of the **cthatstune** command is to change the tunable parameters of the Topology Service subsystem at runtime. When a communication group is created, the Topology Services subsystem is, under normal operating conditions, configured with the default values for these parameters or values that you supply to the **mkcomg** command. You can modify these parameters using the **chcomg** command, as described in "Modifying a communication group's characteristics" on page 73. You can also use the **cthatstune** command to change the parameters directly. The **chcomg** and **cthatstune** commands allow you to change the parameters without restarting the Topology Services subsystem.

For more information about **cthatstune**, **chcomg**, and **mkcomg**, see the *Technical Reference: RSCT for AIX* and *Technical Reference: RSCT for Multiplatforms* guides.

Files and directories

The Topology Services subsystem uses only certain directories, each for a specific purpose.

The Topology Services subsystem uses the following directories:

- **/var/ct/***cluster_name***/log/cthats**, for log files
- **/var/ct/***cluster_name***/run/cthats**, for Topology Services daemon current working directory
- **/var/ct/***cluster_name***/soc/cthats**, for the UNIX domain socket files.

The /var/ct/*cluster_name*/log/cthats (log files):

The **/var/ct/***cluster_name***/log/cthats** directory contains trace output from the Topology Services startup command (**cthats**),

Topology Services daemon (**hatsd**), and NIM.

There are four different log files that are created in this directory: the startup command log, the service version of the daemon trace log, the user version of the daemon trace log, and the NIM trace log. The files, each with the same names on all nodes in the cluster, have the following conventions:

1. The Topology Services log from the **cthats** startup command is:

   cthats.*cluster_name*[.*n*]

   where:

   cluster_name is the name of the cluster to which the node belongs.

   n is a number from 1 to 7 with **cthats**.*cluster_name*.**1** being the most recent instance of the file and **cthats**.*cluster_name*.**7** being the least recent instance.

   The seven most recent instances are kept and older instances are removed.
2. The service version of the log from the **hatsd** daemon is:

   cthats.*DD.HHMMSS.cluster_name*

   where:

   DD is the Day of the Month that this daemon was started.

   HHMMSS is the Hour, Minute, and Second that the daemon was started.

   cluster_name is the name of the cluster (RSCT peer domain) to which the node belongs.

   The contents of this log might be used by IBM Service to help diagnose a problem. The five most recent instances of this file are kept and older instances are removed.
3. The user version of the trace log from the **hatsd** daemon is:

cthats.*DD.HHMMSS.cluster_name.locale*

where:

DD is the Day of the Month that this daemon was started.

HHMMSS is the Hour, Minute, and Second that the daemon was started.

cluster_name is the name of the cluster (RSCT peer domain) to which the node belongs.

locale is the language locale in which the Topology Services daemon was started.

This user version contains error messages that are issued by the **hatsd** daemon. The file provides detailed information that can be used together with the syslog for diagnosing problems.

4. The NIM trace log from the pluggable NIM is:

   nim.cthats.*interface_name***.***nnn*

   where:

   interface_name is the network interface name. For example, **eth0**.

   nnn is a number from 001 to 003

   with **nim.cthats.***interface_name***.001** being the most recent instance of the backup file and **nim.cthats.***interface_name***.003** the oldest instance. The file without the trailing *nnn* is the current NIM trace log.

   The default NIM shipped with Topology Services limits the size of its trace log files to about 200 KB. When the NIM trace log file grows to that limit, the current NIM trace log file is renamed to the most recent back up file and a new NIM trace log file is created. The current and 3 most recent instances of the back up files are kept and the older instances are removed.

The Topology Services daemon limits the size of both the service and user log files to 5,000 lines by default. That limit can be altered by the **cthatstune** command. When the limit is reached, the **hatsd** daemon appends the string '**.bak**' to the name of the current log file and begins a new log file with the same original name. A file that already exists with the '**.bak**' qualifier is removed before the current log is renamed.

The /var/ct/*cluster_name***/run/cthats directory (daemon working files):**

The **/var/ct/***cluster_name***/run/cthats** directory is the current working directory for the Topology Services daemon.

If the Topology Services daemon abnormally terminates, the core dump file is placed in this directory. Whenever the Topology Services daemon starts, it renames any core file to **core**.*DD.HHMMSS.cluster_name*, where:

DD is the Day of the Month that the daemon associated with this core file was started.

HHMMSS is the Hour, Minute, and Second that the daemon associated with this core file was started.

cluster_name is the name of the RSCT peer domain to which the node belongs.

The machines list file is also kept in this directory.

The /var/ct/*cluster_name***/soc/cthats directory (socket files):**

The **/var/ct/***cluster_name***/soc/cthats** directory contains the UNIX domain sockets used for communications between the Topology Services daemon, its clients, and NIMs.

The UNIX domain socket name for communications between the Topology Services daemon and its clients is **server_socket**. The UNIX domain socket name for communications between the Topology Services daemon and NIMs is *NIM_name.pid*, where:

NIM_name is the executable name of the NIM. The name of the default NIM shipped with the Topology Services is **default_ip_nim**.

pid is the PID of the NIM process.

Components on which Topology Services depends

The Topology Services subsystem depends on a number of RSCT components

The Topology Services subsystem depends on the following components:

System Resource Controller (SRC)
A subsystem feature that can be used to define and control subsystems. The Topology Services subsystem is called **cthats**. The subsystem name is used with the SRC commands (for example, **startsrc** and **lssrc**).

Cluster data
For system configuration information established by the configuration resource manager.

UDP/IP and UNIX-domain socket communication
Topology Services daemons communicate with each other using the UDP/IP sockets. Topology Service daemons communicate with client applications and NIMs using UNIX-domain sockets.

Network adapters
Topology Services will form heartbeat rings on the network.

Cluster security services libraries
The Topology Services subsystem uses the Cluster security services libraries (*libct_mss.a* and *libct_sec.a*) to perform message signature and verification.

First Failure Data Capture (FFDC)
When the Topology Services subsystem encounters events that require system administrator attention, it uses the FFDC facility of RSCT to generate entries in an AIX error log on AIX nodes and the System Log on Linux nodes.

Configuring and operating Topology Services

You can perform tasks that affect how the components of the Topology Services subsystem work together to provide those services.

You can perform tasks that affect how the components of the Topology Services subsystem work together to provide the services, such as:

- Setting Topology Services tunables
- Configuring Topology Services
- Initializing the Topology Services daemon
- Operating Topology Services

Attention: Topology Services is controlled by the configuration resource manager. Under normal operating conditions, it should not be necessary to use these Topology Services commands directly. User intervention of Topology Services may cause the configuration resource manager to go down . Exercise caution when operating Topology Services manually.

You can also use the **lssrc** command to display detailed status information about the Topology Services daemon.

Setting Topology Services tunables

The cluster data server stores node and network information, as well as some tunable data.

The following is a list of the attributes and a brief description of each. Many of these tunables can be set using the **mkcomg** or **chcomg** commands (as described in “Creating a communication group” on page 77

and “Modifying a communication group's characteristics” on page 73. You can also use the **cthatstune** command (as described in “The tuning command (cthatstune)” on page 291) to modify Topology Services tunables.

Frequency
Controls how often Topology Services sends a heartbeat to its neighbors. The value is interpreted as the number of seconds between heartbeats. The minimum and default value is 1. On a system with a high amount of paging activity, this number should be kept as small as possible.

Sensitivity
Controls the number of missed heartbeat messages that will cause a Death in Family message to be sent to the Group Leader. Heartbeats are not considered missing until it has been twice the interval indicated by the Frequency attribute. The default sensitivity value is 4.

Pinning
This controls the memory Pinning strategy. **TEXT** causes the daemon to attempt to pin Text pages, **DATA** attempts to pin Data Pages, **PROC** attempts to pin all pages, and **NONE** causes no pages to be pinned. The default is **PROC**.

The following tunables are available only on AIX nodes:

Run_FixedPri
Run the daemon with a fixed priority. Since Topology Services is a real time application, there is a need to avoid scheduling conflicts. A value of 1 indicates that the daemon is running with fixed priority, 0 indicates that it is not.

FixedPriValue
This is the actual fixed priority level that is used. The daemon will accept values greater than or equal to 10. The default is 38.

Log_Length
This is the approximate number of lines that a log file can hold before it wraps. The default is 5000 lines.

The following tunables are available only on Linux nodes:

Fixed Priority
This is the actual fixed priority level to be used. A value of 0 indicates that the daemon is running at the normal priority level. Linux systems allow fixed priority levels from 1 to 99. The higher the priority level, the higher the precedence for the process to run. Topology Services limits the priority level to a range of between 1 and 80. The default level is 30.

Maximum daemon log file length
This is the approximate number of lines that a log file can hold before it wraps. The default is 5000 lines.

On systems with heavy or unusual load characteristics, it might be necessary to adjust the Frequency and Sensitivity settings. See “Operating the Topology Services daemon” on page 296 for more information.

Configuring Topology Services

You may change the default Topology Services configuration options using the **cthatsctrl** command. The **cthatsctrl** command provides a number of functions for controlling the operation of the Topology Services system.

You can use the **cthatsctrl** command to:

- Add or configure the Topology Services subsystem
- Start the subsystem
- Stop the subsystem
- Delete or unconfigure the subsystem

- "Clean" all Topology Services subsystems
- Turn tracing of the Topology Services daemon on or off
- Refresh (read and dynamically reflect a updated configuration) the subsystem.

Adding the Topology Services subsystem:

The **cthatsctrl** command fetches the port number from the cluster data and places it in the **/etc/services** file.

Port numbers are assigned by the configuration resource manager and can be specified when issuing the **mkrpdomain** command. See "Creating a peer domain definition" on page 52 for more information on the **mkrpdomain** command.

The second step is to add the Topology Services daemon to the SRC using the **mkssys** command.

On AIX nodes, a third step is to add an entry in the **/etc/inittab** file so that the Topology Services daemon will be started during boot.

Note that if the **cthatsctrl** add function terminates with an error, you can rerun the command after fixing the problem. The command takes into account any steps that already completed successfully.

Starting and stopping the Topology Services subsystem:

The start and stop functions of the **cthatsctrl** command run the **startsrc** and **stopsrc** commands, respectively. However, **cthatsctrl** automatically specifies the subsystem argument to these SRC commands.

Deleting the Topology Services subsystem:

The delete function of the **cthatsctrl** command removes the subsystem from the SRC, and removes the Topology Services daemon communications port number from **/etc/services**.

On AIX nodes, the delete function also removes the entry from **/etc/inittab**. The delete function does not remove anything from the cluster data, because the Topology Services subsystem might still be configured on other nodes in the cluster.

Tracing the Topology Services subsystem:

The tracing function of the **cthatsctrl** command supplies additional problem determination information, but only when requested by the IBM Support Center.

Because it can slightly degrade the performance of the Topology Services subsystem and can consume large amounts of disk space in the **/var** file system, do not turn tracing on under normal circumstances.

Initializing the Topology Services daemon

Normally, the Topology Services daemon is started by the configuration resource manager when it brings a cluster online.

If necessary, you can start the Topology Services daemon using the **cthatsctrl** command or the **startsrc** command directly. The first part of initialization is done by the startup command, **cthats**. It starts the **hatsd** daemon, which completes the initialization steps.

Understanding the Topology Services daemon initialization process:

During the initialization process, the **cthats** startup command performs certain activities.

During the initialization process, the **cthats** startup command does the following:

1. Determines the number of the local node.
2. Obtains the name of the cluster.
3. Retrieves the **machines.lst** file from the local filesystem, where it was placed by the configuration resource manager. The file has identical contents across the active members of the cluster.
4. Performs file maintenance in the log directory and current working directory to remove the oldest log and rename any core files that might have been generated.
5. Starts the Topology Services **hatsd** daemon.

The daemon then continues the initialization with the following steps:

1. Reads the current machines list file and initializes internal data structures.
2. Initializes daemon-to-daemon communication, as well as client communication.
3. Starts the NIMs.
4. For each local adapter defined, forms a membership consisting of only the local adapter.

The daemon is now in its initialized state and ready to communicate with Topology Services daemons on other nodes. The intent is to expand each singleton membership group formed during initialization to contain as many members as possible. Each adapter has an offset associated with it. Only other adapter membership groups with the same offset can join together to form a larger membership group. Eventually, as long as all the adapters in a particular network can communicate with each other, there will be a single group to which all adapters belong.

Merging all adapters into a single group:

Initially, the subsystem starts out as *N* singleton groups, one for each node.

Each of those daemons is a Group Leader of those singleton groups and knows which other adapters could join the group by the configuration information. The next step is to begin proclaiming to subordinate nodes.

The proclaim logic tries to find members as efficiently as possible. For the first 3 proclaim cycles, daemons proclaim to only their own subnet and, if the subnet is broadcast-capable, that message is broadcast. The result is that, given the previous assumption that all daemons started out as singletons, this would evolve into *M* groups, where *M* is the number of subnets that span this heartbeat ring. On the fourth proclaim cycle, those *M* Group Leaders send proclaims to adapters that are outside of their local subnet. This will cause a merging of groups into larger and larger groups until they have coalesced into a single group.

From the time the groups were formed as singletons until they reach a stabilization point, the groups are considered unstable. The stabilization point is reached when a heartbeat ring has no group changes for the interval of 10 times the heartbeat send interval. Up to that point, the proclaim continues on a 4 cycle operation, where 3 cycles only proclaim to the local subnets, and one cycle proclaims to adapters not contained on the local subnet. After the heartbeat ring has reached stability, proclaim messages go out to all adapters not currently in the group regardless of the subnet to which they belong. Adapter groups that are unstable are not used when computing the node connectivity graph.

Operating the Topology Services daemon

Normal operation of the Topology Services subsystem does not require administrative intervention. The subsystem is designed to recover from temporary failures, such as node failures or failures of individual Topology Services daemons.

Topology Services also provides indications of higher level system failures. However, there are some operational characteristics of interest to system administrators and after adding or removing nodes or adapters, you might need to refresh the subsystem.

Topology Services defaults and limitations:

The maximum node number allowed is 2047. The maximum number of networks it can monitor is 48.

Topology Services is meant to be sensitive to network response and this sensitivity is tunable. However, other conditions can degrade the ability of Topology Services to accurately report on adapter or node membership. One such condition is the failure to schedule the daemon process in a timely manner. This can cause daemons to be late in sending their heartbeats by a significant amount. This can happen because an interrupt rate is too high, the rate of paging activity is too high, or there are other problems. If the daemon is prevented from running for enough time, the node might not be able to send out heartbeat messages and will be considered, incorrectly, to be down by other peer daemons.

Since Topology Services is a real time process, do not intentionally subvert its use of the CPU because you can cause false indications.

On AIX nodes, Topology Services sets all four of the following options to 1 so that the reliable message feature, which utilizes IP source routing, will continue to work:

- **ipsrcroutesend** (default is 1)
- **ipsrcrouterecv** (default is 0)
- **ipsrcrouteforward** (default is 1)
- **nonlocsrcroute** (default is 0)

Disabling any of these network options can prevent the reliable message feature from working properly.

ATTENTION - READ THIS FIRST

The network options to enable IP source routing are set to their default values for security reasons. Since changing them may cause the node to be vulnerable to network attack, system administrators are advised to use other methods to protect the cluster from network attack.

Topology Services requires the IP source routing feature to deliver its data packets when the networks are broken into several network partitions. The network options must be set correctly to enable the IP source routing. On Linux Systems, the Topology Services startup command will set the following options:

IP forward: enable

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Accept Source Routing: enable

```
echo 1 > /proc/sys/net/ipv4/conf/all/accept_source_route
```

```
echo 1 > /proc/sys/net/ipv4/conf/interface/accept_source_route
```

RP Filter: disable

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
```

```
echo 0 > /proc/sys/net/ipv4/conf/interface/rp_filter
```

Dependency on the loopback network interface:

Correct operation of Topology Services depends on the loopback network interface (**lo0**) being defined and in the **up** state.

If the loopback network interface is disabled, Topology Services are unable to transmit Group Services through non-IP devices. Group Services messages are sent through non-IP interfaces when no IP communication paths exist between a specified pair of nodes.

Configuring local adapter state detection

When the local adapter's state is in doubt, such as when heartbeating begins to fail or there are no neighbors available to heartbeat with, the NIM calls an internal library of network monitoring functions called **netmon** to monitor the adapter's health.

If there is no naturally-occurring traffic, the NIM tries to send out "pings" in an attempt to generate traffic and prove the adapter is working. With no user intervention, these pings can only be sent to IP addresses that are known as part of the cluster. This means the smaller the size of the cluster, the fewer targets will be available, and thus the harder it might be for the NIM to verify that a good adapter is working.

See “Pluggable NIMs” on page 289 for more information.

The netmon.cf file:

You can use an optional configuration file called **netmon.cf** to augment the normally-available ping targets with any reachable hosts anywhere on the network that are not already defined to be part of the cluster.

When selecting usable targets, it is important to measure reachability based on the IP addresses that the Topology Services subsystem is monitoring (see “Traditional entries”).

A sample of the **netmon.cf** file, which consists solely of comments about how to populate it, is shipped in this location: **/usr/sbin/rsct/samples/hats/netmon.cf**. To use this file, update it with actual entries and put it in one of these locations:

- In a PowerHA environment: **/usr/es/sbin/cluster/netmon.cf**
- In an RSCT peer domain: **/var/ct/cfg/netmon.cf**

If the **netmon.cf** file exists in one of these paths on a node, every NIM on the node will read this file when Topology Services are started and tries to use the file contents when exercising the network monitor library functions.

The **netmon.cf** file is not controlled by any cluster subsystems. It is not distributed to any other nodes in the cluster because it is added to one of them. It must be put on each node that needs it manually.

Traditional entries:

The traditional format of **netmon.cf** entries is one entry per line. Each entry is an IP address or a host name (short or long). To avoid host name resolution problems, using IP addresses is highly recommended.

Here are some sample **netmon.cf** entries:

```
1.2.3.4                # comments are acceptable -
1.2.3.5                # and are ignored
mynode1.mylab.here.us  # host names can be used,
mynode2                # but are not recommended
2.2.3.4
2.3.4.5
```

A maximum of 32 entries can be specified this way. Any additional lines are ignored. Blank or comment-only lines do not count toward this limit. IP addresses need to be in the appropriate format for their IP version (dotted decimal for IPv4 or colon-separated hexadecimal for IPv6).

At startup time, each NIM reads the file and saves all of the traditional entries, if it does not find any special handling instructions (as described in the following sections). These traditional entries are then used as additional ping targets, beyond whatever the NIM would already have used, any time the local adapter is being checked and appears quiet.

Although traditional entries are used by every NIM, even if the entry is really only reachable by a subset of the adapters on the node, this does not adversely affect adapter state detection. It does not matter how many pings are attempted, or even if any of them succeed, as long as *some* traffic is received by the adapter, to prove it is working before the NIM gives up.

However, it is important to keep in mind which IP addresses are being directly monitored by Topology Services for selection of traditional entries. Generally speaking, Topology Services are given the "boot" addresses to monitor, that is, the addresses that exist on the adapters at boot time.

In high-availability products that use alias IP addresses to keep certain addresses highly available by moving them from adapter to adapter or node to node, the Topology Services subsystem is usually unaware of those addresses. Therefore, a traditional **netmon.cf** entry that is only reachable using one of the "service aliases" and not one of the "boot" addresses is unlikely to help with verifying the local adapter's state.

To be certain which addresses are being monitored on a given node, the subsystem can be queried using **lssrc -ls**.

Handling virtual adapters:

A **!REQD** entry in the **netmon.cf** file indicates special handling is needed that is very different than the traditional **netmon** methods.

A **!REQD** entry has this format:

```
!REQD owner target
```

More than one line identifying the same owner is allowed, and is in fact recommended.

The NIM monitoring the adapter which matches the *owner* field will no longer consider general traffic on the adapter when testing for the adapter's health. No matter how busy the adapter appears to be, it will ping the targets provided (however many **!REQD** entries exist) and *must* receive a successful response to at least one of those pings for the local adapter to be considered "good". For example:

```
# en0 must be able to ping one of these addresses when a
# local health check occurs, or it will be considered down
!REQD en0 1.2.3.4
!REQD en0 1.2.3.5
!REQD en0 1.2.3.6
# All other adapters will continue to use the "traditional"
# netmon method, with these additional ping targets available
2.2.3.4
2.3.4.5
```

Although multiple entries are encouraged, this still restricts the definition of a working adapter from "able to receive any traffic" to "able to ping one of a finite set of targets."

Therefore, this is only recommended in situations where both the following are true:

- General network traffic is not considered a reliable indicator of the adapter's status
- The local adapter's health is still important.

The best example of this would be a single physical box hosting multiple operating system (OS) images with virtual adapters.

It is possible that even if the underlying physical adapters supporting the virtual adapters are lost, and network traffic off of the physical box is no longer possible, the virtual adapters on each of the OS images can still communicate with each other.

From the perspective of any one of the OS images, the adapter is still functional and has connectivity to some network of other nodes, but if a high availability solution is being hosted from one of these OS images, connectivity to something outside the physical box may be important.

To force the virtual adapters to be declared down when the underlying physical adapters are lost (or unplugged), this **netmon.cf** option allows customers to specify targets which are outside of the physical box, ensuring that communication to a broader network than just that local group of virtual images is possible.

To set up these **netmon.cf** entries to properly identify whether a virtual adapter has external network connectivity, keep these points in mind:

- The leading **!REQD** entry must be the beginning of the line (no leading spaces or tabs)
- *owner* can be identified in four ways:
  1. **!ALL**

     The special string **!ALL** can be used to easily match every interface on the node, if desired.
  2. Host name

     Host names are not recommended due to the danger of host name resolution delays. If multiple IP addresses exist on the adapter, the host name must resolve to the address that Topology Services is monitoring in order to be recognized.
  3. Interface name (**en0**, for example)
  4. IP address

     If multiple IP addresses exist on the adapter, the address that Topology Services is monitoring must be used in order to be recognized.
- *target* can be identified in two ways:
  1. Host name

     Host names are not recommended due to the danger of host name resolution delays.
  2. IP address
- Regardless of how the owner and target are specified, the same concerns about "boot" addresses and aliases apply as with "traditional" **netmon.cf** entries. Whatever address Topology Services is using to monitor an owner must be able to reach its target.
- For addressing virtual environments such as described above, using these entries only makes sense if all targets selected are outside of the physical box where the virtual adapter is being served.
- These new entry types can be mixed with traditional entries in **netmon.cf**. NIMs whose adapters match one or more "owners" will use those entries only and ignore any traditional entries. Other NIMs will just use any traditional entries as usual.
- Up to 32 entries can be specified for a given adapter, as identified by *owner*. This is separate from any **!REQD** entries applying to other owners, or traditional entries that might also exist.

  A file can validly have 32 **!REQD** entries per interface for any number of interfaces, plus 32 "traditional" entries if there are any interfaces left that are not specified in a **!REQD** entry.

  Using different ways to identify an owner (IP addresses versus interface names) all counts toward the 32-entry limit. For example, if interface **en0** holds IP address **1.2.3.4**, five owner entries of **en0** and five more owner entries of **1.2.3.4** will count as 10 total entries for that adapter.

  It is recommended that traditional entries are placed at the end of the file, especially if there are many of them. If a NIM reaches the 32-entry limit of traditional entries, it stops reading the configuration file there and will not read one of these new entries if it occurs later in the file.

Special InfiniBand options:

This section applies to the AIX and Linux operating systems only.

If a line is entered in the **netmon.cf** file in one of the following formats, it indicates special handling for InfiniBand adapters only. If any other type of adapter is referenced in this respect, the entry is ignored.

```
!IBQPORT owner

!IBQPORTONLY owner
```

For example:

```
!IBQPORT ib0
!IBQPORTONLY ib1
2.2.3.4
2.3.4.5
```

where:

!IBQPORT
: When verifying the local adapter's health, the physical and logical port states (as seen in commands such as **ibstat**) will be checked first. If either is bad, then the adapter will be declared down. If both are good, then IP traffic will be checked as well, and the adapter's state will be decided based on that.

  Because a check of IP traffic is still done with this option, other netmon.cf entries can still be used to augment whatever ping targets the customer might want the adapter to have.

!IBQPORTONLY
: When verifying the local adapter's health, only the physical and logical port states will be used to make the decision. If both are good the adapter will be considered up; if either is bad then it will be considered down. IP traffic will not be checked in any way.

  Because a check of IP traffic is *not* done with this option, any **netmon.cf** entries related to pinging targets will be ignored by the NIM picking up this entry.

These entry types can be used on normal InfiniBand interfaces, but were primarily aimed at virtual adapters, because the port states of a physical adapter are reflected in any virtual adapters it is serving. This improves monitoring ability for virtual adapters when the state of the underlying physical adapter is considered important.

owner can be identified in four ways:

1. **!ALL**

   The special string **!ALL** can be used to easily match every InfiniBand adapter on the node, if desired.
2. Host name

   Host names are not recommended due to the danger of host name resolution delays. If multiple IP addresses exist on the adapter, the host name must resolve to the address that the Topology Services subsystem is monitoring in order to be recognized.
3. Interface name (**ib0**, for example)
4. IP address

   If multiple IP addresses exist on the adapter, the address that the Topology Services subsystem is monitoring must be used in order to be recognized.

The two entry types are mutually exclusive. If both of them exist and they refer to the same owner, that NIM obey whichever entry was the last one written in the **netmon.cf** file (the last one it picks up to use).

One way multiple entries with the same owner might be used is if the customer wishes most of their InfiniBand adapters to work one way, but a subset of them to work differently. For example:

```
!IBQPORT !ALL
!IBQPORTONLY ib3
```

The result of these two lines would be that all of the NIMs monitoring InfiniBand adapters would pick up the first line, telling them to check the port states and then verify IP traffic. Only the NIM monitoring ib3 would pick up the second line, which would replace what the first line told it to do, and it would only check the port states.

Which clusters and nodes need this file?:

In terms of the traditional entries, there is no absolute rule about whether a cluster does or does not need a **netmon.cf** file.

The primary factor to consider is that the smaller the cluster - meaning the fewer other adapters that are available to help verify any given adapter's state - the more likely a **netmon.cf** file will be needed.

Although there can be exceptions, this means that most two-node clusters will need a **netmon.cf** file at some point, even if they aren't always under conditions where it makes a difference. Three-node clusters are very likely to need one as well.

It is sometimes prudent to consider adapters within the context of their network as well. For example, if two nodes out of a five-node cluster host adapters on a special subnet this is not common to the other three nodes, those adapters might need the benefit of a **netmon.cf** file.

You need to decide whether the entries to handle virtual adapters are needed for a configuration, based on how important it is to recognize whether an adapter can reach the network outside the physical box where it resides, balanced against the reliability of the virtual infrastructure - how likely it is for all paths of communication off the box to be disrupted.

You also need to decide about using the special InfiniBand entries on AIX or Linux. This might be considered useful with any InfiniBand adapter, but is likely to be of greater concern with virtual adapter implementations. Some products exploiting Topology Services with InfiniBand adapters might set one of these values for a customer at configuration time, depending on the needs of the application.

Here are some general notes (unrelated to a specific entry type):

- If this file is needed for a given cluster, it is not necessarily true that every node in the cluster will need to use it, although it is unusual for one node in a cluster to need it while others do not.
- Similarly, its contents do not need to be the same on every node, although that tends to be the common practice since nodes in a cluster usually make use of a common set of adapter types and subnets.

Tuning the Topology Services subsystem

It is necessary for you to decide which settings are suitable for your system.

The default settings for the frequency and sensitivity tunable attributes discussed in “Configuring Topology Services” on page 294 are overly aggressive for clusters that have more than 128 nodes or heavy load conditions. Using the default settings will result in false failure indications. Decide which settings are suitable for your system by considering the following:

- Higher values for the frequency attribute result in lower CPU and network utilization from the Topology Services daemon. Higher values for the product of frequency times sensitivity result in less sensitivity of Topology Services to factors that cause the daemon to be blocked or messages to not reach their destinations. Higher values for the product also result in Topology Services taking longer to detect a failed adapter or node.
- If the nodes are used primarily for parallel scientific jobs, use the settings shown in Table 93 on page 303.

Table 93. Topology Services tunable attribute settings for parallel scientific jobs

Frequency	Sensitivity	Seconds to detect node failure
2	6	24
3	5	30
3	10	60
4	9	72

- If the nodes are used in a mixed environment or for database workloads, use the settings shown in Table 94.

Table 94. Topology Services tunable attribute settings for mixed environments or database workloads

Frequency	Sensitivity	Seconds to detect node failure
2	6	24
3	5	30
2	10	40

- If the nodes tend to operate in a heavy paging or I/O intensive environment, use the settings shown in Table 95.

Table 95. Topology Services tunable attribute settings for heavy paging or I/O intensive environments

Frequency	Sensitivity	Seconds to detect node failure
1	12	24
1	15	30

By default Topology Services uses the settings shown in Table 96.

Table 96. Default Topology Services tunable attribute settings

Frequency	Sensitivity	Seconds to detect node failure
1	4	8

You can adjust the tunable attributes by using the **chcomg** command (as described in "Modifying a communication group's characteristics" on page 73). You can also use the **cthatstune** command. For example, to change the frequency attribute to the value 2 on network **en_net_0** and then refresh the Topology Services subsystem, use the command:

```
cthatstune -f en_net_0:2 -r
```

EtherChannel and IEEE 802.3ad Link Aggregation considerations for AIX

On AIX machines, EtherChannel and IEEE 802.3ad Link Aggregation are network port aggregation technologies that allow several Ethernet adapters to be aggregated together to form a single pseudo Ethernet device.

For example, *ent0* and *ent1* can be aggregated to *ent3*; interface *en3* would then be configured with an IP address. The system considers these aggregated adapters as one adapter. Therefore, IP is configured over them as over any Ethernet adapter. In addition, all adapters in the EtherChannel or Link Aggregation are given the same hardware (MAC) address, so they are treated by remote systems as if they were one adapter.

The main benefit of EtherChannel and IEEE 802.3ad Link Aggregation is that they have the network bandwidth of all of their adapters in a single network presence. In addition, if an adapter fails, the packets are automatically sent on the next available adapter without disruption to existing user connections. The adapter is automatically returned to service on the EtherChannel or Link Aggregation when it recovers.

Details on how to configure EtherChannel and IEEE 802.3ad Link Aggregation are provided in the AIX System Management Guide: Communications and Networks online information.

The link aggregation technologies provide quick detection and recovery from adapter failures. Once a given adapter fails, other adapters which are part of the aggregation will take over IP communication within around 4 seconds.

When the adapter problem is fixed, the adapter gets reactivated into the aggregation. At that point, a disruption of around 8 seconds in IP communication may occur. This disruption is caused by the adapter being declared fit for use by AIX while the switch is still evaluating the new topology. The duration of the disruption may depend on the brand and size of the switch, and may be reduced by configuring the switch not to use Spanning Tree Protocol.

Because adapter failure and recovery may lead to short-term communication outages, RSCT needs to be tuned to allow for a longer adapter detection time. Without tuning, false failure indications may occur during the outages.

The values to be tuned are the Topology Services heartbeat frequency and heartbeat sensitivity. The exact value to be used depends on the length of the communication outage, which itself depends on factors such as adapter type, and brand and size of the switch. A good initial set of values is one that results in detection time around 16 seconds.

It is suggested that experiments be performed to determine how lengthy the outages are for a given system configuration. The experiments should consist of pulling adapter cables and then reconnecting them after a few minutes. If error log entries of type TS_LOC_DOWN_ST or TS_DEATH_TR are generated (assuming that RSCT is running when the experiments are attempted), then this is an indication that the adapter detection tunables need to be increased. To help determine the length of the outages, a sequence such as the following example can be run:

```
while:
  do date
    ping -w1 -c1 <IP address>
    sleep 1
  done
```

The interval during which the packets are lost ("`100% packet loss`" seen in the output) determines for how long communication with the aggregation was not available.

Network Interface Backup (NIB):

EtherChannel Backup is a variant of EtherChannel that is used for high-availability only. EtherChannel Backup allows an aggregated adapter to have a backup. If all adapters that compose the aggregation fail, then communication is switched to the backup adapter until any adapter in the main channel recovers. A variant of it is Network Interface Backup (NIB).

In the NIB mode of operation, there is only 1 adapter in the main channel and a backup adapter. While NIB by itself does not provide better bandwidth than the physical adapter, it can be used to work around switch failures. Usually port aggregation requires all adapters to be connected to the same switch, which makes the switch the single point of failure. By using NIB, and by connecting the primary and backup adapters to different switches, communication will not be lost by the failure of a single switch.

To help detect loss of network reachability (in addition to detecting failures in the adapter and its connection to the switch), NIB allows specifying an address to be pinged. If the given address cannot be reached after a given number of attempts (both specified when NIB is defined), then the current active adapter is considered down, resulting in the backup adapter taking over communication. Setting reasonable values for the Number of Retries option is important to ensure smooth operation of NIB: if the value is not enough to cover the period during which the switch is reconfiguring itself, it is likely that

there will be multiple (false) takeover operations until one of the adapters becomes the owner of the aggregation. Such extra takeover activity makes real (or desired) takeover operations take much longer than intended.

As an initial guideline, setting Number of Retries to 10 should correct the false takeover problem in cases where communication outages are around 8 seconds.

The false takeover scenario can be identified by examining the AIX error log. In case the scenario occurs, entries like the following may appear:

- ECH_PING_FAIL_PRMRY
- ECH_PING_FAIL_BCKP
- GXENT_LINK_DOWN

When Number of Retries is set to an adequate value, then error log entry ECH_CHAN_FAIL may be the only one to be generated.

Since NIB uses a single adapter as primary, an EtherChannel-enabled switch is not required.

Refreshing the Topology Services daemon

In an RSCT peer domain, all refresh operations should occur without user intervention.

You must refresh the Topology Services subsystem before it can recognize a new configuration. However, if you need to manually refresh the Topology Services subsystem, run either the **cthatsctrl** command or the **cthatstune** command, both with the **-r** option on any node in the cluster.

Note that if there are nodes in the cluster that are unreachable with Topology Services active, they will not be refreshed. Also, if the connectivity problem is resolved such that Topology Services on that node is not restarted, the node refreshes itself to remove the old configuration. Otherwise, it will not acknowledge nodes or adapters that are part of the configuration, but not in the old copy of the configuration.

Displaying the status of the Topology Services daemon

You can display the operational status of the Topology Services daemon by issuing the **lssrc** command.

Topology Services monitors the networks that correspond to the communication groups set up by the configuration resource manager. To see the status of the networks, issue the following command on a node that is up:

```
lssrc -ls cthats
```

In response, the **lssrc** command writes the status information to the standard output. The information includes:

- The information provided by the **lssrc -s cthats** command (short form).
- Seven lines for each network for which this node has an adapter and includes the following information:
  - The network name.
  - The network index.
  - The number of defined members, number of adapters that the configuration reported existing for this network.
  - The number of members, number of adapters currently in the membership group.
  - The state of the membership group, denoted by `S` (Stable), `U` (Unstable), or `D` (Disabled).
  - Adapter ID, the address and instance number for the local adapter in this membership group.

  - Group ID, the address and instance number of the membership group. The address of the membership group is also the address of the group leader.
  - Adapter interface name.
  - HB Interval, which corresponds to the **Frequency** attribute in the cluster. This exists both on a per network basis and a default value which could be different.
  - HB Sensitivity, which corresponds to the **Sensitivity** attribute in the cluster. This exists both on a per network basis and a default value which could be different.
  - The total number of missed heartbeats detected by the local adapter, and the total number in the current instance of the group.
  - Two lines of the network adapter statistics.
  - The PID of the NIMs.
- The number of clients connected and the client process IDs and command names.
- Configuration Instance, the Instance number of the Machines List file.
- Whether the daemon is using message authentication. If it is, the version number of the key used for mutual authentication is also included.
- The size of the data segment of the process and the number of outstanding allocate memory without corresponding free memory operations.
- The segments pinned. **NONE**, a combination of **TEXT**, **DATA**, and **STACK**, or **PROC** .
- The size of text, static data, and dynamic data segments. Also, the number of outstanding memory allocations without a corresponding free memory operation.
- Whether the daemon is processing a refresh request.
- Daemon process CPU time, both in user and kernel modes.
- The number of page faults and the number of times the process has been swapped out.
- The number of nodes that are seen as reachable (up) from the local node and the number of nodes that are seen as not reachable (down).
- A list of nodes that are either up or down, whichever list is smaller. The list of nodes that are down includes only the nodes that are configured and have at least one adapter which Topology Services monitors. Nodes are specified in the list using the format:
  - *N1–N2(I1) N3–N4(I2)...*

  where *N1* is the initial node in a range, *N2* is the final node in a range, and *I1* is the increment. For example, 5–9(2) specifies nodes 5, 7, and 9. If the increment is 1 then the increment is omitted. If the range has only one node, only the one node number is specified.

The following is an example of the output from the **lssrc -ls cthats** command on a node:

```
Subsystem         Group            PID     Status
 cthats           cthats           827     active
Network Name   Indx Defd Mbrs St Adapter ID      Group ID
en_net_0       [ 0]    3    2  S 9.114.67.72     9.114.67.73
en_net_0       [ 0] eth0         0x32c37ded      0x32c3907b
HB Interval = 1 secs. Sensitivity = 4 missed beats
Missed HBs: Total: 10 Current Group: 2
Packets sent    : 4706 ICMP 0 Errors: 0 No mbuf: 0
Packets received: 3537 ICMP 0 Dropped: 0
NIM's PID: 884
  1 locally connected Client with PID:
hagsd(   907)
  Configuration Instance = 1244520230
  Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
  Daemon employs no security
  Segments pinned: Text Data Stack.
  Text segment size: 548 KB. Static data segment size: 486 KB.
  Dynamic data segment size: 944. Number of outstanding malloc: 88
```

```
User time 3 sec. System time 1 sec.
Number of page faults: 1245. Process swapped out 0 times.
Number of nodes up: 2. Number of nodes down: 1.
Nodes down: 1
```

The network being monitored in the last example is named **en_net_0**. The **en_net_0** network has 3 adapters defined and 2 of them are members of the group. The group is in stable state. The frequency and sensitivity of this network is 1 second and 4 missing heartbeats respectively. Currently, there is only one client, **hagsd**. The total number of missed heartbeats detected by the local adapter is 10, and the total number in the current instance of the group is 2. All text, data, and stack segments are pinned in the main memory. There are 2 nodes up and 1 node down. The down node is node 1.

Disabled adapter state information

When a network adapter is in the disabled state, the **lssrc** command provides additional state information to identify the reason why the adapter is down.

This state information appears after the adapter interface name in the **lssrc -ls cthats** command output.

Example: The following portion of the output from the **lssrc -ls cthats** command shows where the adapter state information will appear:

```
Network Name   Indx Defd Mbrs St Adapter ID      Group ID
en_net_0       [ 0]    3    0  D 9.114.67.72
en_net_0       [ 0] eth0         adapter_state_information
```

The following are the possible values for *adapter_state_information* and their explanations.

Adapter state
: **Explanation**

Adapter state unknown
: This is the initial value for the adapter state before any determination has been done.

No traffic on adapter
: The adapter has no incoming traffic.

Adapter's interface flags set to down
: The adapter's interface flags have been set to down.

Adapter is misconfigured
: There is a problem with the adapter's configuration, such as a missing or incorrect adapter address.

Broadcast address is misconfigured
: The configured broadcast address is inconsistent with the adapter's IP address and subnet mask.

Adapter is not monitored
: The adapter is intentionally not being monitored.

Adapter has no NIM running
: The adapter has no living network interface module (NIM) associated with it.

Netmon library error
: Indicates an error from the netmon library, which is used to monitor adapter status.

NIM could not bind UDP socket
: The NIM was unable to bind to the UDP socket, possibly due to the port being in use already.

NIM could not open device
: A non-IP NIM was unable to open the device.

The Group Services subsystem

The configuration resource manager uses the Group Services subsystem to provide distributed coordination, messaging, and synchronization among nodes in an RSCT peer domain.

When issuing the **startrpdomain** command to bring a cluster (RSCT peer domain) online, the configuration resource manager, if necessary, starts the Group Services. Under normal operating conditions, it is not necessary for you to directly influence Group Services.

Related concepts:

"An RSCT peer domain" on page 35
An *RSCT peer domain* is a cluster of nodes configured for high availability.

"Group Services subsystem" on page 15
The *Group Services subsystem* (or *Group Services*) provides other RSCT applications and subsystems within an RSCT peer domain with a distributed coordination and synchronization service.

Related tasks:

"Defining preferred nodes for Group Services name server and group leader selection" on page 83
Because the Group Services daemons that act in the name server or group leader roles can consume a significant amount of system resources when it performs those duties, it is sometimes desirable to define the preferred nodes for Group Services to consider (or non-preferred nodes to avoid) when selecting candidates for these roles.

Introducing Group Services

Group Services is a distributed subsystem of the IBM Reliable Scalable Cluster Technology (RSCT) software.

RSCT software provides a set of services that support high availability on your system. Another service included with the RSCT software is the Topology Services distributed subsystem. The Topology Services subsystem is described in "The Topology Services subsystem" on page 286.

The function of the Group Services subsystem is to provide other subsystems with a distributed coordination and synchronization service. These other subsystems that depend upon Group Services are called *client subsystems*. Each client subsystem forms one or more *groups* by having its processes connect to the Group Services subsystem and use the various Group Services interfaces. A process of a client subsystem is called a *GS client*.

A group consists of two pieces of information:

- The list of processes that have joined the group, called the *group membership list*.
- A client-specified *group state value*.

Group Services guarantees that all processes that are joined to a group see the same values for the group information, and that they see all changes to the group information in the same order. In addition, the processes may initiate changes to the group information via *protocols* that are controlled by Group Services.

A GS client that has joined a group is called a *provider*. A GS client that wishes only to monitor a group, without being able to initiate changes in the group, is called a *subscriber*.

Once a GS client has initialized its connection to Group Services, it can join a group and become a provider. All other GS clients that have already joined the group (those that have already become providers) are told as part of a join protocol about the new providers that wish to join. The existing providers can either accept new joiners unconditionally (by establishing a one-phase join protocol) or vote on the protocol (by establishing an n-phase protocol). During a vote, they can choose to approve the protocol and accept the new providers into the group, or reject the protocol and refuse to allow the new providers to join.

Group Services monitors the status of all the processes that are joined to a group. If either the process or the node on which a process is executing fails, Group Services initiates a failure protocol that informs the remaining providers in the group that one or more providers have been lost.

Join and failure protocols are used to modify the membership list of the group. Any provider in the group may also propose protocols to modify the state value of the group. All protocols are either unconditional (one-phase) protocols, which are automatically approved and not voted on, or conditional (n-phase) protocols, which are voted on by the providers.

During each phase of an n-phase protocol, each provider can take application-specific action and **must** vote to approve, reject, or continue the protocol. The protocol completes when it is either approved (the proposed changes become established in the group), or rejected (the proposed changes are dropped).

Group Services components

Several component comprises the Group Services subsystem.

The Group Services subsystem consists of the following components:

Group Services daemon
The central component of the Group Services subsystem.

Group Services API (GSAPI)
The application programming interface that GS clients use to obtain the services of the Group Services subsystem.

Port numbers
TCP/IP port numbers that the Group Services subsystem uses for communications. The Group Services subsystem also uses UNIX domain sockets.

Control command
A shell command that is used to add, start, stop, and delete the Group Services subsystem, which operates under control of the SRC. On Linux, SRC is an RSCT subsystem. On AIX, it is a component of the operating system.

Files and directories
Various files and directories that are used by the Group Services subsystem to maintain run-time data.

The Group Services daemon (hagsd)

The Group Services daemon is the executable file **/usr/sbin/rsct/bin/hagsd**. This daemon runs on each node in the peer domain.

A Group Services client communicates with a Group Services daemon that is running on the same node as the Group Services client. A Group Services client communicates with the Group Services daemon, through the GSAPI software, using a UNIX domain socket. In a PowerHA cluster, before a Group Services client registers with Group Services, it must set the **HA_DOMAIN_NAME** environment variable to the PowerHA cluster name and the **HA_GS_SUBSYS** environment variable to **grpsvcs**, unless PowerHA is running in a CAA environment. In an RSCT peer domain, the **HA_DOMAIN_NAME** and the **HA_GS_SUBSYS** environment variables should *not* be set.

The Group Services API

The Group Services application programming interface (GSAPI) is a shared library that a Group Services client uses to obtain the services of the Group Services subsystem.

This shared library is supplied in two versions: one for non-thread-safe programs and one for thread-safe programs. Table 97 on page 310 gives the path names for each version of the GSAPI shared library on AIX, Linux, Solaris, and Windows nodes.

Table 97. Path names for the GSAPI shared library

	On AIX nodes	On Linux, Solaris, and Windows nodes
Non-thread-safe version	/usr/lib/libha_gs.a	/usr/lib/libha_gs.so
Thread-safe version	/usr/lib/libha_gs_r.a	/usr/lib/libha_gs_r.so

The path names shown in the preceding table are symbolic links to the actual files located in **/usr/sbin/rsct/lib**. For serviceability, these libraries are supplied as shared libraries.

For more information about the GSAPI, see the *RSCT: Group Services Programming Guide*.

Allowing non-root users to use Group Services:

It is possible to allow non-root users to use Group Services.

Do the following to allow non-root users to use Group Services:

1. Create a group named **hagsuser**.
2. Add the desired user IDs to the **hagsuser** group.
3. Stop and restart **cthags** (if it was running before you created the **hagsuser** group).

Users in the created **hagsuser** group can use Group Services.

Port numbers and sockets

The Group Services subsystem uses UDP port numbers and UNIX domain sockets for various communications purposes.

The Group Services subsystem uses several types of communications:

- UDP port numbers for intra-domain communications, that is, communications between Group Services daemons within an operational domain which is defined within the cluster.
- UNIX domain sockets for communication between GS clients and the local Group Services daemon (via the GSAPI).

Intra-domain port numbers:

For communication between Group Services daemons within an operational domain, the Group Services subsystem uses a single UDP port number.

This port number is provided by the configuration resource manager during cluster creation. You supply the port number using the **-g** flag on the **mkrpdomain** command (as described in "Creating a peer domain definition" on page 52).

The Group Services port number is stored in the cluster data so that, when the Group Services subsystem is configured on each node, the port number is fetched from the cluster data. This ensures that the same port number is used by all Group Services daemons in the same operational domain within the cluster.

This intra-domain port number is also set in the **/etc/services** file, using the service name **cthags**. The **/etc/services** file is updated on all nodes in the cluster.

UNIX domain sockets:

UNIX domain sockets are used for communication between GS clients and the local Group Services daemon (through the GSAPI). These are connection-oriented sockets. The socket name used by the GSAPI to connect to the Group Services daemon is **/var/ct/***cluster_name***/soc/hagsdsocket**.

The cthagsctrl control command

The Group Services control command is contained in the executable file **/usr/sbin/rsct/bin/cthagsctrl**.

The purpose of the **cthagsctrl** command is to add (configure) the Group Services subsystem to the cluster. It can also be used to remove the subsystem from the cluster; and start and stop the subsystem. Normally, you will not need to issue this command directly. In fact, in an RSCT peer domain, the configuration resource manager controls the Group Services subsystem, and using this command directly could yield undesirable results. Use this command in an RSCT peer domain only if an IBM service representative instructs you to do so.

For more information, see "Configuring Group Services" on page 312.

Files and directories used by Group Services

Group Services uses several directories on AIX, Linux, Solaris, and Windows nodes.

Table 98 summarizes the directories that the Group Services subsystem uses on AIX, Linux, Solaris, and Windows nodes.

Table 98. Directories used by the Group Services subsystem

Directories for:	On AIX, Linux, Solaris, and Windows nodes
The Group Services daemon (working directory)	**/var/ct/***cluster_name***/run/cthags**
Lock files	**/var/ct/***cluster_name***/lck/cthags**
Log files	**/var/ct/***cluster_name***/log/cthags**
Socket files	**/var/ct/***cluster_name***/soc/cthags**

The Group Services daemon's working directory:

On AIX nodes and Linux nodes, the working directory for the Group Services daemon is **/var/ct/***cluster_name***/run/cthags**.

If the Group Services daemon terminates abnormally, the core dump file is placed in this working directory. Whenever the Group Services daemon starts, it renames any core file to **core.0**. If trace spooling is enabled, the core file is moved to the spooling directory with a time stamp appended to it.

Lock files:

On AIX nodes and Linux nodes, lock files are located in **/var/ct/***cluster_name***/lck/cthags**.

In the lock file directory, **cthags.tid.** is used to ensure a single running instance of the Group Services daemon, and to establish an instance number for each invocation of the daemon.

Log files:

On AIX nodes and Linux nodes, log files are located in the **/var/ct/***cluster_name***/log/cthags** directory. The log file directory contains trace output from the Group Services daemon.

On the nodes, the log files are:

cthags.default.*node-num_inst-num*, which is a text file

trace, which is a binary file

trace.summary, which is a binary file

where:

node-num is the number of the node on which the daemon is running

inst-num is the instance number of the daemon

In an RSCT peer domain, the Group Services trace files are named **trace** and **trace.summary**. When the Group Services daemon is restarted, the previous trace files are copied as **trace.last** and **trace.summary.last**. If there is a core dump file, the trace files are copied as **trace.0** and **trace.summary.0**. The core file will be named **core.0**. If trace spooling is enabled, the core file will be moved to the spooling directory with a time stamp appended to it.

In a PowerHA domain, the Group Services trace files are named **grpsvcs_trace** and **grpsvcs_trace.summary**. When the Group Services daemon is restarted, the previous trace files are copied as **grpsvcs_trace.last** and **grpsvcs_trace.summary.last**. If there is a core dump file, the trace files are copied as **grpsvcs_trace.0** and **grpsvcs_trace.summary.0**.

To convert the binary trace files to text files, use:

```
/usr/sbin/rsct/bin/rpttr -odtic trace trace.summary > trace.total.txt
```

where **trace.total.txt** is the converted file.

Components on which Group Services depends

The Group Services subsystem depends on a number of RSCT components.

The Group Services subsystem depends on the following components:

System Resource Controller (SRC)
: A subsystem that can be used to define and control subsystems. The Group Services subsystem is called **cthags**. The subsystem name is used with the SRC commands (for example, **startsrc** and **lssrc**).

Cluster data
: For system configuration information established by the configuration resource manager.

Topology Services
: A subsystem that is used to determine which nodes in a system can be reached and are running at any given time. Also referred to as the **heartbeat**, the Topology Services subsystem is SRC-controlled. It is called **cthats**. For more information, see "The Topology Services subsystem" on page 286.

UDP/IP and UNIX-domain socket communication
: Group Services daemons communicate with each other using the UDP/IP feature sockets. Topology Service daemons communicate with client applications using UNIX-domain sockets.

First Failure Data Capture (FFDC)
: When the Group Services subsystem encounters events that require system administrator attention, it uses the FFDC facility of RSCT to generate entries in a syslog.

Configuring and operating Group Services

You can perform tasks that affect how the components of the Topology Services subsystem work together to provide those services, such as configuring Group Services and initializing the Group Services daemon. You can also display status information about the Group Services daemon.

Under normal operation, the Group Services subsystem requires no administrative intervention.

Configuring Group Services

Group Services configuration is performed by the **cthagsctrl** command, which is invoked by the configuration resource manager.

Under normal operating conditions, you will not need to directly invoke this command. In fact, doing so could yield undesirable results. In an RSCT peer domain, use this command only if instructed to do so by an IBM service representative.

The **cthagsctrl** command provides a number of functions for controlling the operation of the Group Services system. You can use it to:

- Add (configure) the Group Services subsystem
- Start the subsystem
- Stop the subsystem
- Delete (unconfigure) the subsystem
- Clean all Group Services subsystems
- Turn tracing of the Group Services daemon on or off

Adding the subsystem:

The **cthagsctrl** command fetches the port number from the cluster data.

The second step is to add the Group Services daemon to the SRC using the **mkssys** command.

Note: If the **cthagsctrl** add function terminates with an error, the command can be rerun after the problem is fixed. The command takes into account any steps that already completed successfully.

Starting and stopping the subsystem:

The start and stop functions of the **cthagsctrl** command run the **startsrc** and **stopsrc** commands, respectively. However, **cthagsctrl** automatically specifies the subsystem argument to these SRC commands.

Deleting the subsystem:

The delete function of the **cthagsctrl** command removes the subsystem from the SRC, and removes the Group Services daemon communications port number from **/etc/services**. It does *not* remove anything from the cluster data, because the Group Services subsystem may still be configured on other nodes in the operational domain.

Cleaning the subsystem (AIX only):

On AIX, the clean function of the **cthagsctrl** command performs the same function as the delete function, except in all system partitions. In addition, it removes the Group Services daemon remote client communications port number from the **/etc/services** file.

The clean function does *not* remove anything from the cluster data. This function is provided to support restoring the system to a known state, where the known state is in the cluster data.

Tracing the subsystem:

The tracing function of the **cthagsctrl** command is provided to supply additional problem determination information when it is requested by the IBM Support Center. Normally, tracing should *not* be turned on, because it might slightly degrade Group Services subsystem performance and can consume large amounts of disk space in the **/var** file system.

Initializing the Group Services daemon

Normally, the Group Services daemon is started by the configuration resource manager when it brings a cluster (RSCT peer domain) online. If necessary, the Group Services daemon can be started using the **cthagsctrl** command or the **startsrc** command directly.

During initialization, the Group Services daemon performs the following steps:

1. It gets the number of the node on which it is running. On AIX, the Group Services daemon gets this information from the local peer domain configuration. On Linux, the Group Services daemon gets this information from the cluster definition file which was configured during the RSCT configuration.
2. It tries to connect to the Topology Services subsystem. If the connection cannot be established because the Topology Services subsystem is not running, it is scheduled to be retried every 20 seconds. This continues until the connection to Topology Services is established. Until the connection is established, the Group Services daemon writes an error log entry periodically and no clients may connect to the Group Services subsystem.
3. It performs actions that are necessary to become a daemon. This includes establishing communications with the SRC subsystem so that it can return status in response to SRC commands.
4. It establishes the Group Services domain, which is the set of nodes in the cluster.

   At this point, one of the GS daemons establishes itself as the GS nameserver. For details, see "Establishing the GS nameserver."

   Until the domain is established, no GS client requests to join or subscribe to groups are processed.
5. It enters the main control loop.

   In this loop, the Group Services daemon waits for requests from GS clients, messages from other Group Services daemons, messages from the Topology Services subsystem, and requests from the SRC for status.

Establishing the GS nameserver:

The Group Services subsystem must be able to keep track of the groups that its clients want to form. To do this, it establishes a GS nameserver within the domain. The GS nameserver is responsible for keeping track of all client groups that are created in the domain.

As described in "Defining preferred nodes for Group Services name server and group leader selection" on page 83, certain nodes can be defined as preferred nodes when selecting a node to fill the GS nameserver role.

To ensure that only one node becomes a GS nameserver, Group Services uses the following protocol:

1. When each daemon is connected to the Topology Services subsystem, it waits for Topology Services to tell it which nodes are currently running in the peer domain and which of these nodes, if any, are preferred nodes.
2. Based on the input from Topology Services, each daemon finds the lowest-numbered candidate node in the node list, as follows:
   - If there is at least one preferred node in the node list, the candidate node is the lowest-numbered node that is also a preferred node.
   - If there are no preferred nodes in the node list, the candidate node is the lowest-numbered node.
3. The daemon compares its own node number to the candidate node and does one of the following:
   - If the node the daemon is on is the candidate node, the daemon waits for all other running nodes to nominate it as the GS nameserver.
   - If the node the daemon is on is not the candidate node, it sends nomination messages to the lowest-numbered candidate node periodically, initially every 5 seconds.
4. Once all running nodes have nominated the GS nameserver-to-be and a coronation timer (about 20 seconds) has expired, the nominee sends an insert message to the nodes. All nodes must acknowledge this message. When they do, the nominee becomes the established GS nameserver, and it sends a commit message to all of the nodes.
5. At this point, the Group Services domain is established, and requests by clients to join or subscribe to groups are processed.

Note that this protocol is in effect when all nodes are being booted simultaneously, such as at initial system power-on. It is often the case, however, that a Group Services daemon is already running on at

least one node and is already established as the GS nameserver for the domain. In that case, the GS nameserver waits only for Topology Services to identify the newly running nodes. The GS nameserver will then send proclaim messages to those nodes, directing them to nominate it as nameserver. Once those nodes then nominate the GS nameserver, the GS nameserver executes one or more insert protocols to insert the newly-running nodes into the domain.

During recovery from a GS nameserver failure, Group Services selects a new nameserver candidate node, as follows:

- If there is at least one preferred node in the node list, the new candidate node is the next node in the list that is also a preferred node.
- If there are no preferred nodes in the node list, the new candidate node is the next node in the list.

Be aware that, even when preferred nodes are defined, it is possible for a non-preferred node to be selected as the GS nameserver if none of the preferred nodes are available.

Group Services initialization errors:

The Group Services subsystem creates error log entries to indicate severe internal problems. For most of these, the best response is to contact the IBM Support Center.

However, if you get a message that there has been no heartbeat connection for some time, it could mean that the Topology Services subsystem is not running.

To check the status of the Topology Services subsystem, issue the **lssrc -l -s cthats** command. If the response indicates that the Topology Services subsystem is inoperative, try to restart it using the **startrpdomain** or **startrpnode** command. If you are unable to restart it, call the IBM Support Center.

Group Services daemon operation

Normal operation of the Group Services subsystem requires no administrative intervention.

The subsystem normally recovers from temporary failures, such as node failures or failures of Group Services daemons, automatically. However, there are some operational characteristics that might be of interest to administrators:

- The maximum number of groups to which a GS client can subscribe or that a GS client can join is equivalent to the largest value containable in a signed integer variable.
- The maximum number of groups allowed within a domain is 65,535.
- These limits are the theoretical maximum limits. In practice, the amount of memory available to the Group Services daemon and its clients will reduce the limits to smaller values.

Establishing a GS group leader:

When a provider first attempts to join a group, it causes Group Services to create the group and designate a node as the group leader.

As described in “Defining preferred nodes for Group Services name server and group leader selection” on page 83, certain nodes can be defined as preferred nodes when selecting a node to fill the GS group leader role.

Group Services selects the group leader, as follows:

- If the node on which the first-joining provider resides is a preferred node, then that node becomes the GS group leader. The group is created with the joining node as the first member of the group.
- If the node on which the first-joining provider resides is *not* a preferred node, then:

– If the GS nameserver node *is* a preferred node, then the nameserver node also becomes the GS group leader. An MG Group is created on the nameserver node and that node is inserted as the first member of the group. The node that originated the join request can then join as the next member of the group.
– If the GS nameserver node *is not* a preferred node, then the node that originated the join request becomes the GS group leader. The group is created with the joining node as the first member of the group.

Upon creation of a group, even when preferred nodes are defined, the GS group leader will reside on a non-preferred node when neither the first-joining node nor the GS nameserver node are preferred nodes. Subsequently, if a preferred node joins the group, Group Services does not switch the group leader role to that node. Such node changes can only occur during recovery of a failing group leader.

During recovery from a GS group leader failure, Group Services selects a new group leader, as follows:

- If there is at least one preferred node in the group member list, the new group leader is the next node in the list that is also a preferred node.
- If there are no preferred nodes in the group member list, the new group leader is the next node in the list.

Thus, during GS recovery, it is possible for the GS group leader to move from a preferred node to a non-preferred node, or from a non-preferred node to a preferred node, based on the group membership at the time.

Displaying the status of the Group Services daemon:

You can display the operational status of the Group Services daemon by issuing the **lssrc** command.

Enter the command as follows:

lssrc -l -s cthags

In response, the **lssrc** command writes the status information to standard output. The information includes:

- The information provided by the **lssrc -s cthags** command (short form)
- The number of currently connected clients and their process IDs
- The status of the Group Services domain
- The node number on which the GS nameserver is running
- Statistics for client groups with providers or subscribers on this node.

Note that if the **lssrc** command times out, the Group Services daemon is probably unable to connect to the Topology Services subsystem. For more information, see "Group Services initialization errors" on page 315.

This sample output is from the **lssrc -l -s cthags** command on a node in the cluster:

```
Subsystem         Group            PID     Status
 cthags             cthags             11938   active
 4 locally-connected clients.  Their PIDs:
 21344(sample_test1) 17000(sample_test3) 18200(rmcd)
 HA Group Services domain information:
 Domain established by node 9.
 Number of groups known locally: 2
                    Number of   Number of local
 Group name         providers   providers/subscribers
 WomSchg_1              5           1           1
 rmc_peers              7           1           0
```

In this domain, the GS nameserver is on node 9 of the system.

If a GS nameserver has not yet been established, the status indicates that the domain is not established. Similarly, if the GS nameserver fails, the status shows that the domain is recovering. Both of these conditions should clear in a short time. If they do not and the Topology Services subsystem is active, call the IBM Support Center.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte 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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM 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 supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this

one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
U.S.A.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All 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 the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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.

Privacy policy considerations

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 the 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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

INFINIBAND, InfiniBand Trade Association, and the INFINIBAND design marks are trademarks and/or service marks of the INFINIBAND Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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.

Red Hat, the Red Hat "Shadow Man" logo, and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc., in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

Special characters

A

B

C

D

E

F

G

T

U

V

Printed in USA