Reliable Scalable Cluster Technology

IBM

Administration Guide

SA22-7889-11

Reliable Scalable Cluster Technology

IBM

Administration Guide

SA22-7889-11

Note

Before using this information and the product it supports, read the information in “Notices” on page 275.

Twelfth Edition (October 2006)

This edition applies to:
- version 5, release 2 of IBM AIX 5L for POWER (product number 5765-E62) with the 5200-08 Technology Level
- version 5, release 3 of IBM AIX 5L for POWER (product number 5765-G03) with the 5300-05 Technology Level
- version 1, release 6 of IBM Cluster Systems Management (CSM) for Linux on Multiplatforms (product number 5765-E88)
- version 1, release 6 of CSM for Linux on POWER (product number 5765-G16)
- version 2, release 1 of IBM Tivoli System Automation for Multiplatforms (product number 5724-M00)

and to all subsequent releases and modifications, until otherwise indicated in new editions. Vertical lines (|) in the left margin indicate technical changes to the previous edition of this book.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for your comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States and Canada): 1+845+432-9405
FAX (Other Countries)
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet: mhvrcfs@us.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this book

This book describes various component subsystems of IBM Reliable Scalable Cluster Technology (RSCT). On AIX®, the RSCT components are included as part of the AIX 5L™ operating system. The RSCT components are also available as part of various Linux-based products such as IBM® Cluster Systems Management (CSM) for Linux® and IBM Tivoli® System Automation for Multiplatforms. This book describes:

- 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*.
- The basics of cluster security services which are used by other RSCT components and other cluster products for authentication. This book describes some 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.

RSCT is a component of the following:
- AIX 5L
- Cluster Systems Management (CSM) for Linux
- IBM Tivoli System Automation for Multiplatforms

Who should use this book

This book 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.

Conventions and terminology used in this book

Conventions

Table 1 describes the typographic conventions used in this book.

Table 1. Typographic conventions

Typographic convention	Usage
bold	**Bold** words or characters represent system elements that you must use literally, such as: command names, file names, flag names, and path names.
`constant width`	Examples and information that the system displays appear in `constant-width` typeface.

Table 1. Typographic conventions (continued)

Typographic convention	**Usage**
italic	*Italicized* words or characters represent variable values that you must supply. *Italics* are also used for book titles, for the first use of a glossary term, and for general emphasis in text.
{ *item* **}**	Braces indicate required items.
[*item* **]**	Brackets indicate optional items.
*item***...**	Ellipses indicate items that can be repeated.
|	1. In the left margin of the book, vertical lines indicate technical changes to the information. 2. In synopsis statements, vertical lines are used as *pipe* characters.

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

Terminology

This book uses the terminology conventions shown in Table 2:

Table 2. Terminology

Term	**Usage**
HPS	A shorthand notation for the *High Performance Switch,* which works in conjunction with a specific line of IBM System p™ servers

See the “Glossary” on page 279 for definitions of some of the other terms that are used in this book.

Prerequisite and related information

The core Reliable Scalable Cluster Technology (RSCT) publications are:

- *RSCT: Administration Guide*, SA22-7889, provides an overview of the RSCT components and describes how to:
  - Create and administer RSCT peer domains.
  - Manage and monitor resources using the resource monitoring and control (RMC) subsystem.
  - Administer cluster security services for RSCT peer domains and CSM management domains.
- *RSCT: Diagnosis Guide*, SA23-2202, describes how to diagnose and resolve problems related to the various components of RSCT. This book is a companion volume to *RSCT: Messages*, which lists the error messages that may be generated by each RSCT component. While *RSCT: Messages* describes the appropriate user responses to messages that are generated by RSCT components, this book contains additional and more detailed diagnostic procedures.

- *RSCT: Messages*, GA22-7891, lists the error messages that may be generated by each RSCT component. For each message, this manual provides an explanation of the message, and describes how you should respond to it.
- *RSCT for AIX 5L: Technical Reference*, SA22-7890, and *RSCT for Linux: Technical Reference*, SA22-7893, provide detailed reference information about all of the RSCT commands, daemons, files, and scripts.

In addition to these core RSCT publications, the library contains the following publications of interest:

- *RSCT: RMC Programming Guide and Reference*, SA23-1346, describes the resource monitoring and control application programming interface (RMC API). This book is intended for programmers who want to create applications that use the RMC API to connect to the RMC subsystem to leverage its resource management and monitoring capabilities.
- *RSCT: Group Services Programming Guide and Reference*, SA22-7888, contains information for programmers who want to write new clients that use the group services subsystem's application programming interface (GSAPI) or who want to add the use of group services to existing programs. This book is intended for programmers of system management applications who want to use group services to make their applications highly available.
- *RSCT: LAPI Programming Guide*, SA22-7936, provides conceptual, procedural, and reference information about the low-level application programming interface (LAPI). LAPI is part of the AIX implementation of RSCT only; it is not available with RSCT for Linux. LAPI is a message-passing API that provides optimal communication performance on the IBM High Performance Switch (HPS), which works in conjunction with a specific family of IBM System p servers.
- *RSCT for AIX 5L: Managing Shared Disks*, SA22-7937, describes the shared disk management facilities of IBM eServer Cluster 1600 server processors — the optional virtual shared disk and recoverable virtual shared disk components of RSCT for AIX 5L. These components are part of the AIX implementation of RSCT only; they are not available with RSCT for Linux. This book describes how you can use these components to manage cluster disks to enable multiple nodes to share the information they hold. The book includes an overview of the components and explains how to plan for them, install them, and use them to add reliability and availability to your data storage.

For access to all of the RSCT documentation, refer to the **IBM Cluster information center**. This Web site, which is located at **http://publib.boulder.ibm.com/infocenter/clresctr**, contains the most recent RSCT documentation in HTML and PDF formats. The **Cluster information center** also includes an *RSCT Documentation Updates* file, which contains documentation corrections and clarifications, as well as information that was discovered after the RSCT books were published. Check this file for pertinent information (about required software patches, for example).

The current RSCT books and earlier versions of the library are also available in PDF format from the **IBM Publications Center** Web site, which is located at **http://www.ibm.com/shop/publications/order**. It is easiest to locate a manual in the **IBM Publications Center** by supplying the manual's publication number. The publication number for each of the RSCT books is listed after the book title in the preceding list.

Using LookAt to find message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. You can use LookAt from the following locations to find IBM message explanations:

- The Internet. You can access IBM message explanations directly from the LookAt Web site:

  www.ibm.com/eserver/zseries/zos/bkserv/lookat
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer, Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

How to send your comments

Your feedback is important in helping to provide accurate, high-quality information. If you have any comments about this book or any other RSCT documentation:

- Go to the **IBM Cluster Information Center** home page at:

  http://publib.boulder.ibm.com/infocenter/clresctr

  Click on the **Contact us** link to go to our feedback page, where you can enter and submit your comments.
- Send your comments by e-mail to: **mhvrcfs@us.ibm.com**

  Include the book title and order number, and, if applicable, the specific location of the information about which you have comments (for example, a page number, table number, or figure number).
- Fill out one of the forms at the back of this book and return it by mail, by fax, or by giving it to an IBM representative.

Chapter 1. What is RSCT?

Reliable Scalable Cluster Technology, or RSCT, is a set of software components that together provide a comprehensive clustering environment for AIX and Linux. 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**, which provide the security infrastructure that enables RSCT components to authenticate the identity of other parties.
- **Topology Services subsystem**, which, on some cluster configurations, provides node and network failure detection.
- **Group Services subsystem**, which, on some cluster configurations, provides cross-node/process coordination.

What are management domains and peer domains?

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

You configure a set of nodes for manageability using the Clusters Systems Management (CSM) product as described in *IBM Cluster Systems Management: Administration Guide*. The set of nodes configured for manageability is called a *management domain* of your cluster.

You configure a set of nodes for high availability using RSCT's Configuration resource manager. The set of nodes configured for high availability is called an RSCT *peer domain* of your cluster. For more information, refer to Chapter 3, "Creating and administering an RSCT peer domain," on page 17.

Table 3 lists the characteristics of the two domain types—management domains and peer domains—that can be present in your cluster.

Table 3. Characteristics of management domains and peer domains

Management domain characteristics	**Peer domain characteristics**
Established and administered by CSM.	Established and administered by RSCT's Configuration resource manager.

Table 3. Characteristics of management domains and peer domains (continued)

Management domain characteristics	**Peer domain characteristics**
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.	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.
Processor architecture and operating system are heterogeneous.	Processor architecture and operating system are heterogeneous. Starting with RSCT version 2.3.2.0, peer domain nodes can run either AIX or Linux. AIX nodes will support any processor architecture supported by the AIX operating system. The supported Linux distributions are detailed in "Supported Linux distributions for RSCT 2.4.6.0" on page 15. (Note that products designed to run in a peer domain may not support the same heterogeneous environment as RSCT. Please refer to the specific exploiter's documentation for information on supported processor architecture and operating systems.)
The RMC subsystem and core resource managers are used by CSM to manage cluster resources. CSM also provides an additional resource manager — the Domain resource manager.	The RMC subsystem and core resource managers are used to manage cluster resources.
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.

What is RMC?

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 (where the infrastructure is used by the Configuration resource manager) and management domains (where the infrastructure is used by CSM). 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, IP address 9.117.7.21, and so on. 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*.

What are the 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, refer to Chapter 4, "Managing and monitoring resources using RMC and resource managers," on page 63.

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 program products (such as CSM, which contains the Domain Management resource manager).

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

The following are the RSCT core resource managers:

- **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.
- **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. Refer to Chapter 3, "Creating and administering an RSCT peer domain," on page 17 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, refer to "Basic resource monitoring" on page 88 and "Advanced resource monitoring" on page 105.
- **File System resource manager**, which manages file systems.
- **Host resource manager**, which manages resources related to an individual machine.

- **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 Chapter 6, “Understanding and administering the Storage resource manager,” on page 183.
- **Sensor resource manager**, which provides a means to create a single user-defined attribute to be monitored by the RMC subsystem. For more information, refer to “Creating event sensor commands for monitoring” on page 127.
- **CIM resource manager**, which enables you to use RMC to query system configuration through Common Information Model (CIM) classes. For more information, refer to “Querying and monitoring CIM properties and associations” on page 129.
- **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, refer to Chapter 5, “Controlling access to root commands and scripts,” on page 153.

For more information about RMC and the core resource managers, refer to Chapter 4, “Managing and monitoring resources using RMC and resource managers,” on page 63.

What are the cluster security services?

The cluster security services are used by RSCT applications and components 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 and is discussed in “Managing user access to resources using RMC ACL files” on page 80.

Cluster Security Services uses **credential 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.

Credential based authentication involves the use of a third party that both the client and the server trust. For this release, only Host Based Authentication is supported, but other security mechanisms may be supported in the future. In the case of Host Based Authentication, the trusted third party is the UNIX® operating system. This method of authentication is used between RSCT and its client applications (such as CSM), and also by the configuration resource manager during the creation and addition of nodes to an RSCT peer domain.

For more information about the cluster security services, refer to Chapter 7, “Understanding and administering cluster security services,” on page 205.

What are Topology Services?

The Topology Services subsystem is used within an RSCT peer domain to provide other RSCT applications and subsystems 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 this 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, connectivity messages are sent around all heartbeat rings. Each ring forwards its connectivity messages to other rings so that all nodes can construct a connectivity graph. This graph is used by the reliable messaging service to determine the route to use when delivering a message to a destination node.

For more information about Topology Services, refer to Chapter 8, “The Topology Services subsystem,” on page 235.

What are Group Services?

The Group Services subsystem is used within an RSCT peer domain to provide other RSCT applications and subsystems 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.

For more information about Group Services, refer to Chapter 9, “The Group Services subsystem,” on page 253.

What are IBM Virtual shared disks and IBM Recoverable virtual shared disks?

IBM Virtual shared disks and IBM Recoverable virtual shared disks are subsystems of the AIX implementation of RSCT. These RSCT subsystems are provided as part of the AIX operation system and are not available in the Linux implementation of RSCT.

- IBM Virtual shared disk is an RSCT subsystem that lets application programs that are running on different nodes of an RSCT peer domain access a raw logical volume as if it were local at each of the nodes. Each virtual shared disk corresponds to a logical volume that is actually local at one of the nodes, which is called the *server node*. The Virtual shared disk subsystem routes I/O requests from the other nodes, called *client nodes*, to the server node and returns the results to the client nodes.

The I/O routing is done by the Virtual shared disk device driver that interacts with the AIX Logical Volume Manager (LVM). The device driver is loaded as a kernel extension on each node. Thus, raw logical volumes can be made globally accessible.

- The IBM Recoverable virtual shared disk (RVSD) is an RSCT subsystem that provides recoverability of your virtual shared disks if a node, adapter, or disk failure occurs. The RVSD subsystem manages your virtual shared disks and, when an error is detected, will automatically switch disk access to an active node. Recovery is transparent to applications and there is no disruption of service except for a slight delay while takeover occurs.

IBM Virtual shared disks and IBM Recoverable virtual shared disks are implemented as an RMC Resource Manager (the Virtual Shared Disk Resource Manager) which provides a command line interface for configuring and managing virtual shared disks.

For details about IBM Virtual shared disks and IBM Recoverable virtual shared disks, including complete administrative information for these subsystems, refer to *Reliable Scalable Cluster Technology for AIX 5L: Managing Shared Disks*. For complete syntax information on the VSD commands, refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference*.

What is the low-level application programming interface (LAPI)?

The low-level application programming interface (LAPI) is a component of the AIX implementation of RSCT. LAPI is provided as part of the AIX operation system and is not available in the Linux implementation of RSCT.

LAPI is a message-passing API that provides a one-sided communication model. In this model, one task initiates a communication operation to a second task. The completion of the communication does not require the second task to take a complementary action. RSCT LAPI provides optimal communication performance on an IBM eServer™ pSeries® High Performance Switch (pSeries HPS). PSSP LAPI provides optimal communication performance on the SP™ Switch2.

The LAPI library provides basic operations to “put” data to and “get” data from one or more virtual addresses of a remote task. LAPI also provides an active message infrastructure. With active messaging, programmers can install a set of handlers that are called and run in the address space of a target task on behalf of the task originating the active message. Among their other uses, these handlers can be used to dynamically determine the target address (or addresses) where data from the originating task must be stored. You can use this generic interface to customize LAPI functions for your environment.

Some of LAPI’s other general characteristics include:
- Flow control
- Support for large messages
- Support for generic non-contiguous messages
- Non-blocking calls
- Interrupt and polling modes
- Efficient exploitation of switch functions

- Event monitoring support (to simulate blocking calls, for example) for various types of completion events

LAPI is meant to be used by programming libraries and by power programmers for whom performance is more important than code portability.

For complete conceptual, procedural, and reference information about this RSCT component, refer to *Reliable Scalable Cluster Technology for AIX 5L: LAPI Programming Guide*.

What is the System Resource Controller (SRC)?

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 (Topology Services, Group Services, Cluster Security Services, and so on) run under the SRC. On AIX, the SRC is, like the RSCT components, part of the operating system. For the Linux implementation 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. Under normal circumstances, you should not explicitly start or stop the RSCT subsystems. However, when following certain troubleshooting procedures, you may be instructed to use the SRC commands **startsrc** and **stopsrc** to start and stop RSCT subsystems. You can also use the command **lssrc** to list the status of RSCT systems.

Chapter 2. RSCT installation and software verification

The AIX implementation of RSCT is included as part of the AIX 5L operating system. The Linux implementation of RSCT is included with a variety of Linux-based products such as IBM Cluster System Management (CSM) for Linux and IBM Tivoli System Automation for Multiplatforms.

This documentation applies to RSCT version 2.4.6.0 for AIX 5L (version 5.3) and Linux and RSCT 2.3.10.0 for AIX (version 5.2).

RSCT installation verification on AIX nodes

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

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

Output should be similar to the following:

```
Fileset                      Level  State  Type  Description (Uninstaller)
---------------------------------------------------------------------------
rsct.basic.hacmp           2.4.6.0    C     F    RSCT Basic Function (HACMP/ES
                                                 Support)
rsct.basic.rte             2.4.6.0    C     F    RSCT Basic Function
rsct.basic.sp              2.4.6.0    C     F    RSCT Basic Function (PSSP
                                                 Support)
rsct.compat.basic.hacmp    2.4.6.0    C     F    RSCT Event Management Basic
                                                 Function (HACMP/ES Support)
rsct.compat.basic.rte      2.4.6.0    C     F    RSCT Event Management Basic
                                                 Function
rsct.compat.basic.sp       2.4.6.0    C     F    RSCT Event Management Basic
                                                 Function (PSSP Support)
rsct.compat.clients.hacmp  2.4.6.0    C     F    RSCT Event Management Client
                                                 Function (HACMP/ES Support)
rsct.compat.clients.rte    2.4.6.0    C     F    RSCT Event Management Client
                                                 Function
rsct.compat.clients.sp     2.4.6.0    C     F    RSCT Event Management Client
                                                 Function (PSSP Support)
rsct.core.auditrm          2.4.6.0    C     F    RSCT Audit Log Resource
                                                 Manager
rsct.core.errm             2.4.6.0    C     F    RSCT Event Response Resource
                                                 Manager
rsct.core.fsrm             2.4.6.0    C     F    RSCT File System Resource
                                                 Manager
rsct.core.gui              2.4.6.0    C     F    RSCT Graphical User Interface
rsct.core.hostrm           2.4.6.0    C     F    RSCT Host Resource Manager
rsct.core.lprm             2.4.6.0    C     F    RSCT Least Privilege Resource
                                                 Manager
rsct.core.rmc              2.4.6.0    C     F    RSCT Resource Monitoring and
                                                 Control
rsct.core.sec              2.4.6.0    C     F    RSCT Security
rsct.core.sensorrm         2.4.6.0    C     F    RSCT Sensor Resource Manager
rsct.core.sr               2.4.6.0    C     F    RSCT Registry
rsct.core.utils            2.4.6.0    C     F    RSCT Utilities
rsct.exp.cimrm             2.4.6.0    C     F    CIM Resource Manager
rsct.opt.storagerm         2.4.6.0    C     F    RSCT Storage Resource Manager
rsct.opt.saf.amf           2.4.6.0    C     F    RSCT SAF Resource Manager
```

If the RSCT components are installed, ensure that they are at a version that applies to this documentation. This documentation applies to RSCT versions 2.3.10.0 and 2.4.6.0. If you are using RSCT specifically in conjunction with an exploiter of the technology (such as CSM for AIX 5L), you'll want to make sure that this is the version of RSCT required by the exploiter. You should also be aware that not all of the RSCT file sets are required by every RSCT exploiter. Refer to the specific RSCT exploiter's documentation for information on RSCT version and file set requirements.

If you discover you need a later version of this or other RSCT documentation, refer to the **IBM Cluster Information Center**. This Web site is located at publib.boulder.ibm.com/clresctr and always contains the most recent RSCT documentation in PDF and HTML formats. RSCT documentation is also available in PDF format from the **IBM Publications Center** Web site located at www.ibm.com/shop/publications/order.

The RSCT for AIX 5L file sets are described in Table 4.

Table 4. RSCT for AIX 5L file sets

This file set...	Contains...
rsct.basic.rte	• configuration resource manager • group services • topology services
rsct.core	RSCT core components, including: • audit log resource manager • event response resource manager (ERRM) • file system resource manager • host resource manager • resource monitoring and control (RMC) subsystem • cluster security services • sensor resource manager • least privilege resource manager • system registry • miscellaneous utilities
rsct.exp	RSCT Expansion Pack. The RSCT Expansion Pack contains the CIM resource manager. This file set is part of the AIX 5L 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 file sets off the AIX 5L Expansion Pack. The file sets for the Pegasus CIM server are: sysmgt.pegasus.cimserver sysmgt.pagasus.osbaseproviders
rsct.vsd	virtual shared disk and recoverable virtual shared disk subsystems
rsct.lapi	low-level application programming interface (LAPI)
rsct.opt.storagerm	storage resource manager
rsct.opt.saf	• Service Availability Forum (SAF) shared client library • SAF resource manager (IBM.SAFRM) • miscellaneous tools

If entering the **lslpp** command as described above reveals that needed RSCT file sets 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 file set as shown in the preceding table.

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

Notes:

1. If installing the CIM resource manager, note that IBM does not ship a **.bff** (Backup File Format) file in the RSCT expansion pack and so the installation is from the rsct.exp.cimrm image. Since the installation is from an image, the rsct.exp.cimrm fileset will be automatically committed. Remove rsct.exp.cimrm first before rejecting rsct.
2. The file set rsct.opt.storagerm, containing the storage resource manager, was introduced in RSCT 2.4.3.0 and 2.3.7.0. Please note that this file set will not be installed automatically with RSCT 2.4.5.0 or 2.3.9.0 PTF updates, but can be explicitly installed when the PTF updates are applied. If it is, be aware that in order to reject the PTF updates, you will need to uninstall the file set first.
3. Versions of the C++ Runtime Library for AIX prior to version 7.0.0.1 may cause the configuration resource manager to abnormally end. To avoid this, you should obtain PTF U800738 / U800739 in order to update to the C++ Runtime Library for AIX version 7.0.0.1. The PTF is contained in the file **xlc.rte.aix5.oct2004.ptf.tar.Z**, which is available at http://www.ibm.com/support/docview.wss?uid=swg24008213.
4. The file set rsct.opt.saf, containing the SAF shared client library and SAF resource manager, was introduced in RSCT 2.4.6.0 and 2.3.10.0. Normally, you should not modify these files as they are of little value without a SAF client and HA manager exploiting them.

RSCT adapter support for AIX

The following adapters are supported by RSCT for the AIX operating system.

- 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 (AIX 5.3 with the 5300-03 Recommended Maintenance Package, or later, running on a 64-bit Power 4 or greater node is required)
- GX Dual-port 4x IB HCA adapter over InfiniBand (AIX 5.3 with the 5300-03 Recommended Maintenance Package, or later, running on an IBM System p5™ 505 node is required)
- 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 HACMP™ configuration)

RSCT installation verification on Linux nodes

The Linux implementation of RSCT is shipped with a number of products that exploit the technology. The RSCT file sets should be installed by following the specific exploiter's installation procedure. Table 5 lists the packages that are required on the target node before installing RSCT.

Table 5. Packages required on the target node before installing RSCT

Library	**Package**
Standard C Library	glibc
Standard C++ Library	libstdc++
Compatibility standard C++ Library on Red Hat EL 4	compat-libstdc++-33-3.2.3
Compatibility standard C++ Library on SLES 10	compat-libstdc++-5.0.7

For Red Hat EL 4 64-bit kernel (x86_64 and ppc64), there are two RPM packages for compat-libstdc++-33-3.2.3. One is 32-bit, and the other is 64-bit. RSCT requires the 32-bit RPM by default but, if you plan to use the RSCT 64-bit library, 64-bit compat-libstdc++-33-3.2.3 is also required.

To verify that RSCT has been installed on a Linux node, enter:

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

Output should be similar to the following:

```
src-1.3.0.1-0
rsct.core.utils-2.4.6.0-0
rsct.basic-2.4.6.0-0
rsct.core-2.4.6.0-0
```

If the RSCT components are installed, ensure that they are at a version that applies to this documentation. This documentation applies to RSCT for Linux version 2.4.6.0. If you discover you need a later version of this or other RSCT documentation, refer to the **IBM Cluster Information Center**. This Web site is located at publib.boulder.ibm.com/clresctr and always contains the most recent RSCT documentation in PDF and HTML formats. RSCT documentation is also available in PDF format from the **IBM Publications Center** located at www.ibm.com/shop/publications/order.

Table 6 describes the RSCT for Linux RPM packages. In the RPM package names, *platform* will be `i386`, `ppc`, `s390`, `ppc64`, or `x86_64`.

Table 6. RSCT for Linux RPM Packages

This RPM package...	**Contains...**
rsct.basic-2.4.6.0-0.*platform*.rpm	• configuration resource manager • group services • topology services
rsct.core-2.4.6.0-0.*platform*.rpm	RSCT core components, including: • resource monitoring and control (RMC) • audit log resource manager • event response resource manager (ERRM) • file system resource manager • host resource manager • cluster security services • least privilege resource manager • system registry

Table 6. RSCT for Linux RPM Packages (continued)

This RPM package...	Contains...
rsct.core.cimrm-2.4.6.0.*platform*.rpm	CIM resource manager (where available)
rsct.core.utils-2.4.6.0-0.*platform*.rpm	miscellaneous utilities
rsct.opt.storagerm-2.4.6.0-0.*platform*.rpm	storage resource manager
src-1.3.0.1-0.*platform*.rpm	system resource controller
rsct.64bit-2.4.6.0-0.*platform*.rpm	RSCT 64-bit library for 64-bit kernel (except SLES8 ppc64)
rsct.opt.saf.amf-2.4.6.0-0.*platform*.rpm	• Service Availability Forum (SAF) shared client library • SAF resource manager (IBM.SAFRM) • miscellaneous tools

If entering the **rpm** command as described above reveals that needed RSCT RPM packages are not installed, you can install them from the RSCT exploiter's installation media. You should refer to the RSCT exploiter's documentation for installation instructions.

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

1. Install the system resource controller by entering:
   ```
   rpm -i src-1.3.0.1-0.platform.rpm
   ```
2. Install the RSCT utilities by entering:
   ```
   rpm -i rsct.core.utils-2.4.6.0-0.platform.rpm
   ```
3. Install the RSCT core components by entering:
   ```
   rpm -i rsct.core-2.4.6.0-0.platform.rpm
   ```
4. Install the RSCT basic components by entering:
   ```
   rpm -i rsct.basic-2.4.6.0-0.platform.rpm
   ```
5. Optionally, install the RSCT CIM resource manager component by entering:
   ```
   rpm -i rsct.core.cimrm-2.4.6.0.platform.rpm
   ```
6. Optionally, install the RSCT storage resource manager component by entering the command shown below. Note that 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-2.4.6.0-0.platform.rpm
   ```
7. Optionally, install the RSCT Service Availability Forum resource manager (SAF) and the SAF shared client library.
   ```
   rpm -i rsct.opt.saf.amf-2.4.6.0-0.platform.rpm
   ```

If entering the **rpm** command as described in the preceding instructions reveals that previous versions of RSCT RPM packages are installed, you could upgrade RSCT using the **rpm** command.

- If your system does not have the rsct64bit package installed, you can use the following command to upgrade RSCT:
  ```
  rpm -Fvh src-1.3.0.1-0.platform.rpm rsct.core.utils-2.4.6.0-0.platform.rpm
    rsct.core-2.4.6.0-0.platform.rpm rsct.basic-2.4.6.0-0.platform.rpm
    (rsct.core.cimrm-2.4.6.0.platform.rpm rsct.opt.storagerm-2.4.6.0.platform.rpm
    rsct.opt.saf.amf-2.4.6.0-0.platform.rpm)
  ```

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

  ```
  rpm -Fvh src-1.3.0.1-0.platform.rpm rsct.core.utils-2.4.6.0-0.platform.rpm
    rsct.core-2.4.6.0-0.platform.rpm rsct.basic-2.4.6.0-0.platform.rpm
    (rsct.core.cimrm-2.4.6.0.platform.rpm rsct.opt.storagerm-2.4.6.0.platform.rpm
    rsct.opt.saf.amf-2.4.6.0-0.platform.rpm) rsct.64bit-2.4.6.0-0.platform.rpm
  ```

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

If you wish to uninstall RSCT, please note that the packages must be uninstalled in a specific sequence (as shown in the following instructions). If there is any exploiter that has dependency on RSCT, the **rpm** command will not allow you to uninstall the RSCT packages.

1. If the rsct64bit package was installed, uninstall it by entering:
   ```
   rpm -e rsct.64bit
   ```
2. If the SAF resource manager is installed, uninstall it by entering the following command:
   ```
   rpm -e rsct.opt.saf.amf
   ```
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:
   ```
   rpm -e rsct.core.cimrm
   ```
5. Uninstall the RSCT basic components by entering:
   ```
   rpm -e rsct.basic
   ```
6. Uninstall the RSCT core components by entering:
   ```
   rpm -e rsct.core
   ```
7. Uninstall the RSCT utilities by entering:
   ```
   rpm -e rsct.core.utils
   ```
8. Uninstall the system resource controller by entering:
   ```
   rpm -e src
   ```

The Linux distributions supported by this version of RSCT are described in "Supported Linux distributions for RSCT 2.4.6.0" on page 15. Please check your RSCT exploiter's documentation to see if that particular product also supports a particular distribution.

If you are installing RSCT on:
- Red Hat EL 3.0 on AMD-64, refer to the additional instructions in "Kernel requirement for Red Hat EL 3.0 on AMD-64."
- SUSE Linux Enterprise Server 9 (SLES 9), refer to the additional instructions in "Kernel update requirement for SUSE SLES 9 Linux on all platforms" on page 15.

Kernel requirement for Red Hat EL 3.0 on AMD-64

On Red Hat EL 3 on AMD-64, RSCT requires Update 2. Without Red HAT EL 3 Update 2 installed, RSCT may be unstable in cluster mode.

Kernel update requirement for SUSE SLES 9 Linux on all platforms

On SUSE Linux Enterprise Server 9 (SLES 9), RSCT requires SLES 9 updates if your system:

- has SLES 9 GA-level installed,
- the compat service is in effect for the group database,
- and the **/etc/groups** file directs the use of NIS.

On a SLES 9 system, the lookup order of the group database is controlled by the Name Service Switch (NSS). The NSS configuration file may specify the compat service for the group database. When the compat service is specified for the group database, lookup requests are processed by searching the local **/etc/groups** file, and special entries within the **/etc/groups** file can direct the use of the Network Information Service (NIS).

The Name Service Switch (NSS) configuration file is **/etc/nsswitch.conf**. The compat service may be specified explicitly for the group database in this file. If the file does not exist or does not contain an entry for the group database, the compat service is used by default for group database lookups. See the **nsswitch.conf**(5) man page for details. When the compat service is in effect for group database lookups, the following entry in **/etc/groups** directs the use of Network Information Service (NIS):

```
+:::
```

The SLES 9 updates are required because program exceptions may otherwise occur when routines in the C library are called to search the group database while the compat service is in effect for the group database and the **/etc/groups** file directs the use of NIS. Because program exceptions can occur if SLES 9 updates are not installed, some RSCT daemons will not function properly.

Supported Linux distributions for RSCT 2.4.6.0

Table 7 lists the Linux distributions and hardware platforms that are supported by RSCT version 2.4.6.0.

Table 7. Supported Linux distributions for RSCT 2.4.6.0

	Hardware platform						
	System x™			**System p**			
Linux distribution	**x86**	**AMD-64**	**xBlade**	**Power 4, 5**	**JS20 Blade**	**System z™**	**System i™**
CentOS 4	supports 32-bit distribution	not supported	supports 32-bit distribution	not supported	not supported	not supported	not supported
Red Hat EL 3.0 (support indicated includes all three members of the Red Hat EL family — AS, WS, and ES)	supports 32-bit distribution	supports 32-bit and 64-bit distributions	supports 32-bit distribution	supports 32-bit and 64-bit distributions	supports 32-bit and 64-bit distributions	supports 32-bit and 64-bit distributions	not supported
Red Hat EL 4.0 (support indicated includes all three members of the Red Hat EL family — AS, WS, and ES)	supports 32-bit distribution	supports 64-bit distribution	supports 32-bit distribution	supports 64-bit distribution	supports 64-bit distribution	not supported	not supported
Scientific Linux 4	supports 32-bit distribution	not supported	supports 32-bit distribution	not supported	not supported	not supported	not supported

Table 7. Supported Linux distributions for RSCT 2.4.6.0 (continued)

	Hardware platform						
	System x™			**System p**			
Linux distribution	**x86**	**AMD-64**	**xBlade**	**Power 4, 5**	**JS20 Blade**	**System z™**	**System i™**
SUSE LINUX Enterprise Server 9 (SLES 9)	supports 32-bit distribution	supports 64-bit distribution	supports 32-bit distribution	supports 64-bit distribution	supports 64-bit distribution	supports 32-bit and 64-bit distributions	supports 64-bit distribution
SUSE LINUX Enterprise Server 10 (SLES 10)	supports 32-bit distribution	supports 64-bit distribution	supports 32-bit distribution	supports 64-bit distribution	supports 64-bit distribution	not supported	not supported

RSCT adapter support for Linux

The following adapters are supported by RSCT for the Linux operating system.

- 10/100 Ethernet adapter
- 1 GB Ethernet adapter
- 1 GB Myrinet switch adapter
- Topspin adapter over InfiniBand (AIX 5.3 with the 5300-03 Recommended Maintenance Package, or later, running on a 64-bit Power 4 or greater node is required)
- GX Dual-port 4x IB HCA adapter over InfiniBand (AIX 5.3 with the 5300-03 Recommended Maintenance Package, or later, running on an IBM System p5 505 node is required)
- Channel bonding configuration support (an arrangement in which two or more network interfaces on a host computer are combined for redundancy or increased throughput)

Chapter 3. Creating and 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

What is 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 by CSM 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. (Refer to Chapter 7, "Understanding and administering cluster security services," on page 205 for more information.)
- the Topology Services subsystem for node/network failure detection. Generally, the peer domain's use of this subsystem will be transparent to you. (Refer to Chapter 8, "The Topology Services subsystem," on page 235 for more information.)
- the Group Services subsystem for cross node/process coordination. Generally, the peer domain's use of this subsystem will be transparent to you. (Refer to Chapter 9, "The Group Services subsystem," on page 253 for more information.)
- the Resource Monitoring and Control subsystem for coordination between the various RSCT subsystems. Generally, the peer domain's use of this subsystem will be transparent to you. However, you can use RMC to monitor the peer domain. (Refer to Chapter 4, "Managing and monitoring resources using RMC and resource managers," on page 63 for more information.)

What is the 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. Refer to "What can I do using configuration resource manager commands?" on page 20 for more information.

What are 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.

For more information, refer to “Working with communication groups” on page 40.

What is quorum?

Quorum refers to the minimum numbers of 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*.

What is 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.

What is 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.

What is 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 is when a peer domain is inadvertently divided into two or more sub-domains.

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 sub-domain 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 sub-domain are no longer aware of nodes in any other sub-domain. 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 sub-domain has operational quorum and can continue operating, thus becoming the peer domain. Usually, the sub-domain with the majority of nodes will have operational quorum.

How the configuration resource manager uses a tie breaker to determine operational quorum: After domain partitioning, it is usually the sub-domain with the majority of nodes that will have operational quorum. However, sometimes there is a tie in which multiple sub-domains 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 *tie breaker* to determine which sub-domain has operational quorum. A *tie breaker* is an RMC resource defined by the configuration resource manager that specifies how tie situations should be resolved. It is the tie-breaker that determines which sub-domain will have operational quorum and so will survive, and which sub-domain will be dissolved.

For more information, refer to "Determining how the configuration resource manager will resolve tie situations when calculating operational quorum" on page 52.

How the configuration resource manager uses critical resource protection methods: When a sub-domain that has critical resources loses quorum, the configuration resource manager uses a *critical resource protection method* on each node of the sub-domain to ensure that critical resources will not be corrupted. A *critical resource protection method* is simply software that determine how the configuration resource manger will respond when quorum is lost in a sub-domain. A critical resource protection method will also be used on a node whose configuration resource manager, group services, or topology services daemon hangs. There are a number of critical resource protection methods defined by the configuration resource manager. You can specify a critical resource protection method for the entire peer domain or specify one to be used on just one particular node. The critical resource protection methods do such things as halt the system, reset and reboot the system, and so on.

For more information, refer to "Setting the critical resource protection method for a peer domain or a node in a peer domain" on page 49.

What are quorum types?

A peer domain's quorum type specifies how startup quorum, configuration quorum, and operational quorum will be calculated for the peer domain. The quorum types are:

Normal

Normal mode, which is the default for an AIX/Linux cluster. In this mode:

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

Quick Quick startup mode, which is useful for large clusters. In this mode:

```
StartupQuorum = 1
ConfigQuorum = N/2 + 1
OpQuorum = Majority + TieBreaker
```

Override
Override mode. Available only for OS/400® environments and the default for such environments. In this mode:

```
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 and the default for such environments. In this mode:

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

What can I do using configuration resource manager commands?

Table 8 outlines the tasks you can perform using configuration resource manager commands.

Table 8. Tasks you can perform using configuration resource manager commands

To perform this task...	Use these configuration resource manager commands...	For more information, refer to:
Create a peer domain	1. The **preprpnode** command to prepare the security environment on each node that will participate in the peer domain. 2. The **mkrpdomain** command to create a new peer domain definition. 3. The **startrpdomain** command to bring the peer domain online.	“Creating a peer domain” on page 25
Add nodes to an existing peer domain	1. The **preprpnode** command to prepare the security environment on the new node. 2. The **addrpnode** command to add the node to a peer domain. 3. The **startrpnode** command to bring the node online.	“Adding nodes to an existing peer domain” on page 32
Take a peer domain node offline	The **stoprpnode** command	“Taking a peer domain node offline” on page 36
Take a peer domain offline	The **stoprpdomain** command	“Taking a peer domain offline” on page 37
Remove a node from a peer domain	The **rmrpnode** command	“Removing a node from a peer domain” on page 38
Remove a peer domain	The **rmrpdomain** command	“Removing a peer domain” on page 39
List communication groups	The **lscomg** command	“Listing communication groups” on page 41

Table 8. Tasks you can perform using configuration resource manager commands (continued)

To perform this task...	Use these configuration resource manager commands...	For more information, refer to:
Modify a communication group's characteristics (Topology Services' tunables)	the **chcomg** command 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 the underlying network supports it).	"Modifying a communication group's characteristics" on page 42
Manually configure communication groups *(not necessary under normal circumstances; only to be exercised in unavoidable situations)*	the **chcomg** command to modify a communication group's network interface.	"Modifying a communication group's network interface" on page 44
	the **mkcomg** command to create a communication group.	"Creating a communication group" on page 45
	the **rmcomg** command to remove a communication group.	"Removing a communication group" on page 47

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

- modify topology services and group services parameters, and
- determine how the configuration manager responds to domain partitioning to prevent corruption of critical data.

For more information see "Modifying Topology Services and Group Services parameters" on page 47 and "Determining how your system responds to domain partitioning and subsystem daemon failure" on page 49.

When describing how to perform these administrative tasks, the command examples do not necessarily contain a description of all of the command options. For complete information about any of the commands, refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or to *Reliable Scalable Cluster Technology for Linux: Technical Reference.* If you encounter error messages while trying to perform the tasks, refer to *Reliable Scalable Cluster Technology: Messages* for recovery information.

Prerequisites and restrictions to using configuration resource manager commands

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

- The following packages are required. On AIX, these are available as part of the base AIX operating system. On Linux, these packages are shipped with the products (such as CSM) that use the RSCT technology, and should have been installed as part of the product's installation procedure.
  – rsct.core
  – rsct.basic
  – rsct.core.utils
  – rsct.core.sec (required for AIX nodes only)

- All nodes 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 network interfaces you plan to include in the peer domain, and that are in the same subnet, must be configured to use the same MTU (maximum transmission unit) size. This is important because MTU size differences between interfaces in the same subnet can result in packet loss. For more information, see Appendix A, "RSCT network considerations," on page 263.

Supported RSCT versions

RSCT Peer Domain is officially supported by RSCT with a version number of 2.2.1.20 or higher. Although it was possible to create an RSCT Peer Domain with an earlier version (RSCT 2.2.1.10), that version is not officially supported. Nodes running RSCT 2.2.1.10 should **not** be added to a Peer Domain created with RSCT 2.2.1.20 or a later version.

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

```
lslpp -l rsct*
```

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

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

RSCT Support for VIPA

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 address. These routes are needed even if there is no application on the nodes that communicates using the VIPA address. 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, refer to *AIX 5L System Management Guide: Communications and Networks*, SC23-4909.

For information on whether VIPA is supported by a particular RSCT exploiter, refer to 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.

Migration

| There are several considerations to be aware of before attempting to migrate an
| RSCT peer domain (RPD) and update the active RSCT version to a new level.

Migration considerations

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

Avoiding domain partitioning when migrating from RSCT 2.2.1.*x* or 2.3.0.*x*

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

Before migrating an individual node running RSCT 2.2.1.*x* or 2.3.0.*x*, take the node offline using the **stoprpnode** command (as described in "Taking a peer domain node offline" on page 36). 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 35).

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 (using the **startrpdomain** command as described in "Bringing the peer domain online" on page 29) from a node running the higher level of RSCT.

PTF rejection can result in loss of cluster data

When a node is upgraded from a version prior to RSCT 2.3.3.0, the registry information will be saved and converted to a new registry format. If an upgraded node is downgraded with PTF rejection, the new registry will be replaced by the old saved registry. If this happens, any cluster data committed after the upgrade will be lost.

Ensuring that network interfaces are included in communication groups when migrating from versions prior to 2.4.1.0/2.3.5.0

In versions of RSCT prior to 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 migrating from a version of RSCT prior to 2.4.1.0/2.3.5.0, be aware that RSCT will not be able to tell if network interfaces that had been shut down (and so were excluded from liveness checks performed by the communication groups) are later activated. For this reason, after migrating from a version of RSCT prior to 2.4.1.0/2.3.5.0, you should manually set the HeartbeatActive persistent attribute of IBM.NetworkInterface resources that had been shut down. After migrating to the new version of RSCT:

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 will be 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 HeartbeatActive attribute of any of the IBM.NetworkInterface resources is 0, this indicates that the network interface is excluded from the peer domain's communication groups. To specify that the network interface should be included in the peer domain's communication groups, set the HeartbeatActive attribute of the 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 migrating an RSCT cluster from one level to the next level, we strongly recommended that you take the corresponding node or nodes offline in the RSCT peer domain. Once the migration is complete, you can bring the node or nodes back 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, we strongly recommended that you take the RSCT peer domain offline, and then bring it back online after the migration is complete.

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

A fix introduced in RSCT 2.3.3 may 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 will not be recognizable to RSCT 2.3.3 (and later releases) and the domain will not come online. Peer domains that were created under RSCT 2.3.2 in the more distant past (that is, prior to 2006) should not be affected by this change and should 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 creating a new domain.
- If you have a recently-created peer domain under RSCT 2.3.2 that you need to migrate to a newer release of RSCT, contact the IBM Support Center for assistance in devising a unique migration strategy.

Migrating a peer domain

In order to complete the migration of a peer domain and update the active RSCT version to a new level, you must enter the **runact** command as shown below. This command should be run only after all the nodes defined in a peer domain are 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 the upgraded nodes have an RSCT version higher than the active version (RSCTActiveVersion), the new minimum RSCT version across all nodes is determined and becomes the new active version of the peer domain.

To complete the migration of a peer domain:

1. Upgrade nodes defined in a peer domain to a later version.
2. After you have upgraded all the nodes defined in a peer domain, make sure that more than half of the nodes are online. If not, then bring nodes online to meet the criteria.

3. Execute 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 the command is run before all the nodes are 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 executing the command again.

Creating a peer domain

Before creating a peer domain, make sure you have reviewed the information in "Prerequisites and restrictions to using configuration resource manager commands" on page 21.

To configure nodes into an RSCT peer domain, you need to:
- Prepare the initial security environment on each node that will be in the peer domain using the **preprpnode** command.
- Create a new peer domain definition by issuing the **mkrpdomain** command.
- Bring the peer domain online using the **startrpdomain** command.

In a peer domain, processor architecture and operating system are heterogeneous. Starting with version 2.3.2.0 of RSCT, peer domain nodes can run either AIX or Linux. AIX nodes will support any processor architecture supported by the AIX operating system. The supported Linux distributions are detailed in "Supported Linux distributions for RSCT 2.4.6.0" on page 15. (Please note, however, that products designed to run in a peer domain may not support the same heterogeneous environment as RSCT. Please refer to the specific exploiter's documentation for information on supported processor architecture and operating systems.)

Preparing the initial security environment on each node that will participate in the peer domain

Before you can create your peer domain using the **mkrpdomain** command (described in "Creating a new peer domain" on page 27), you first need to run the **preprpnode** command to establish the initial trust between each node that will be 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, you should refer to "Guarding against address and identify spoofing when transferring public keys" on page 223. If you are not sure if your network is secure enough, consult with a network security specialist to see if you are at risk.

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 where 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, say 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 new peer domain” on page 27). 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. For more information about public keys and trusted host list files, refer to Chapter 7, “Understanding and administering cluster security services,” on page 205.
- modifying the local node’s RMC Access Control List (ACL) to enable access to its resources from the originator node. For more information about RMC ACL files, refer to “Managing user access to resources using RMC ACL files” on page 80.

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. You may also want to use the **-k** flag if you know the originator node and the local node have already been configured by CSM as part of the same management domain. In this case, the necessary public key transfer has already occurred. 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 more information on security issues related to the automatic transferal of public keys, refer to Chapter 7, "Understanding and administering cluster security services," on page 205.

For complete syntax information on the **preprpnode** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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.

Creating a new peer domain

The **mkrpdomain** command creates a new peer domain definition. A peer domain definition consists of:

- a peer domain name
- the list of nodes included in that peer domain
- the UDP port numbers to be used for Topology Services and Group Services daemon to daemon communication

For example, say you want to establish a peer domain with three nodes, and the nodes are identified by the DNS names *nodeA*, *nodeB*, and *nodeC*. Say also that, when you issued the **preprpnode** command from the nodes that will make up your peer domain, you determined that *nodeA* would be the originator node. To create a peer domain named *ApplDomain*, you would, from *nodeA*, issue the command:

```
mkrpdomain ApplDomain nodeA nodeB nodeC
```

The characters used for your domain name are limited to the ASCII characters A-Z, a-z, 0-9, . (period), and _ (underscore). The above command creates the peer domain definition *ApplDomain* consisting of the nodes *nodeA*, *nodeB*, and *nodeC*.

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

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

The configuration resource manager will at this time create the communication group definitions needed to later enable liveness checks (known as *heartbeating* in Topology Services) between the nodes of a peer domain. The configuration resource manager will attempt to automatically form a communication group 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 (described in "Bringing the peer domain online" on page 29), the configuration resource manager will supply the communication group definition information to Topology Services. For more information on Topology Services, refer to Chapter 8, "The Topology Services subsystem," on page 235.

Since, in the preceding commands, a quorum type was not specified, a default quorum type will be used to calculate startup quorum, configuration quorum, and operational quorum. The default quorum type will depend on your environment. For most clusters, the default quorum type will be “Normal”. In OS/400 environments, the default will be “Override”. In environments with the IBM TotalStorage SAN File System, the default will be “SANFS”. For a description of the quorum types and how startup quorum, configuration quorum, and operational quorum are calculated for each type, refer to “What are quorum types?” on page 19.

To specify a quorum type, you can use the **-Q** flag followed by an integer or name indicating the quorum type. The quorum types are described in “What are quorum types?” on page 19. On the **mkrpdomain** command, you can specify the quorum type to be one of the following:

- 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 will not need to explicitly set the quorum type to either of these values.

For example, to specify quick startup mode, which is useful for large clusters, you could specify:

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

or

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

When starting a peer domain (as described in “Bringing the peer domain online” on page 29), you can override the quorum type to specify a different one for calculating startup quorum. You can also modify the quorum type as described in “Changing a peer domain’s quorum type” on page 39.

If the **mkrpdomain** command fails on any node, it will, by default, fail for all nodes. You can override this default behavior using the **-c** flag. You might want to use this flag, for example, when creating larger peer domain configurations. If you are creating a peer domain consisting of a large number of nodes, the chances that the **mkrpdomain** command would fail on any one is greater. In such a case, you probably would not want the operation to fail for all nodes based on a single node failing. You would therefore enter:

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

Since, in the preceding commands, port numbers were not specified for Topology Services and Group Services daemon to daemon communication, the default port numbers (port 12347 for Topology Services and port 12348 for Group Services) will be used. You can override these defaults using the **mkrpdomain** command’s **-t** flag (to specify the Topology Services port) or **-g** flag (to specify the Group Services port). Any unused port in the range 1024 to 65535 can be assigned. For example:

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

For many of its operations on a peer domain, the configuration resource manager needs to 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 the configuration resource manager will use is 128. If more remote connections are needed than there are threads available, the configuration resource manager will wait for threads to become available in order to connect with other remote nodes. For very large clusters, performance improvements may be realized by increasing the maximum number of threads the configuration resource manager will use to perform parallel operations. The maximum number of threads is called the *fanout value* and can be specified using the **mkrpdomain** command's **-m** flag.

For example, to specify a fanout value of 900:

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

The fanout value specified on the **mkrpdomain** command will be applied to all subsequent operations in the peer domain, unless explicitly overridden or reset. You can:

- override the fanout value when starting a peer domain using the **startrpdomain** command (described in "Bringing the peer domain online"). The fanout value specified on the **startrpdomain** command, however, applies to that startup operation only.
- reset the fanout value by modifying the Fanout attribute of the IBM.PeerNode resource class. You can use the **chrsrc** command to modify the Fanout attribute. For example, to specify a fanout value of 1000, you would do the following from a node that is online in the peer domain:
  1. Set the management scope to the RSCT peer domain scope:
     ```
     export CT_MANAGEMENT_SCOPE=2
     ```
  2. Issue the following **chrsrc** command:
     ```
     chrsrc -c IBM.PeerNode Fanout=1000
     ```

For complete syntax information on the **mkrpdomain** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Once you have created your peer domain definition, you can bring the peer domain online.

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, simply 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 (as described in "What are quorum types?" on page 19) 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 (as described in "Creating a new peer domain" on page 27). 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 that quick startup mode should be used 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 at this time 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 41). 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 42. To modify network interfaces in the communication group, refer to “Modifying a communication group’s network interface” on page 44.

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
```

Output will be similar to the following:

```
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. Results are similar to the following.

```
Name                      OpState                    RSCTVersion
nodeA                     online                     2.2.1.20
nodeB                     online                     2.2.1.20
nodeC                     online                     2.2.1.20
nodeD                     offline                    2.2.1.20
nodeE                     offline                    2.2.1.20
```

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. The only time you need to explicitly set the CT_MANAGEMENT_SCOPE environment variable is 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 the *Reliable Scalable Cluster Technology: Diagnosis Guide*). 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. These commands are described in "Taking individual nodes of a peer domain, or an entire peer domain, offline" on page 36.

For complete syntax information on the **startrpdomain** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Adding nodes to an existing peer domain

"Creating a peer domain" on page 25 describes the initial setup of a peer domain. Once you have created a peer domain, you can add new nodes to it. To add a node to a peer domain, you need to:

- Prepare the security environment on the node using the **preprpnode** command
- Add the node to the peer domain definition using the **addrpnode** command
- Bring the node online in the peer domain using the **startrpnode** or **startrpdomain** command

Preparing the security environment on the node

Before you can add a node to a peer domain using the **addrpnode** command (described in "Adding the node to the peer domain" on page 34), you first need to 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.

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 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 just the Configuration Manager group leader) 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   2.3.3.0
nodeB   Online   2.3.3.0
nodeC   Online   2.3.3.0
```

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. A Group Leader is a Topology Services and Group Services term for a coordinating node of Configuration Manager group. Although the operation of the Topology Services and Group Services subsystems should be transparent to you, they are used by a peer domain for distributed coordination and synchronization. For more information on Topology Services and Group Services, refer to Chapter 8, "The Topology Services subsystem," on page 235 and Chapter 9, "The Group Services subsystem," on page 253.

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
```

Results will be similar to the following. 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 2003

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. Specifying the Group Leader node eliminates the need to specify the other nodes in the peer domain.

```
preprpnode node8
```

If you have chosen, for security reasons, to manually transfer the public keys, you need to 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. You may also want to use the **-k** flag if you know the originator node and local node have already been configured by CSM as part of the same management domain. In this case, the necessary public key transfer has already occurred. 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 information on security issues related to the automatic transferal of public keys, refer to "Guarding against address and identify spoofing when transferring public keys" on page 223.

For complete syntax information on the **preprpnode** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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

Adding the node to the peer domain

When you initially set up an RSCT peer domain (described in "Creating a peer domain" on page 25), you use the **mkrpdomain** command to create the initial peer domain definition. To now add one or more nodes to that existing peer domain definition, you use the **addrpnode** command, passing it the IP address or DNS name of the node you wish to add. Keep in mind, however, that any change to the online cluster definition requires a *configuration quorum* of (*n*/2)+1 nodes (where *n* is the number of nodes defined in the cluster) to be active. In other words, you can not change an online cluster definition unless a majority of the nodes are online in the domain.

To add the node whose DNS name is *nodeD* to a peer domain, issue the following command from a node in the peer domain:

```
addrpnode nodeD
```

You can also add multiple nodes to the peer domain definition. You can do this either by listing them all on the command line:

```
addrpnode nodeD nodeE
```

Or else you can, using the **-f** flag, specify the name of a file that lists the node names:

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

The configuration resource manager will at this time modify the communication group definitions needed later to extend liveness checks (Topology Services' "heartbeating") to the new nodes. When you issue the **startrpnode** command (described in "Bringing the node online in the peer domain"), the configuration resource manager will supply the modified communication group definition information to Topology Services. For more information on communication groups, refer to "Working with communication groups" on page 40. For more information on Topology Services, refer to Chapter 8, "The Topology Services subsystem," on page 235.

For complete syntax information on the **addrpnode** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Once you have added a node to an existing peer domain definition, you can bring the node online.

Bringing the node online in the peer domain

The **startrpnode** command brings an offline node online in the current peer domain. To see 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                     2.2.1.20
nodeB                    online                     2.2.1.20
nodeC                    online                     2.2.1.20
nodeD                    offline                    2.2.1.20
nodeE                    offline                    2.2.1.20
```

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 will not be brought online in the new peer domain, but will instead remain 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     2.2.1.20              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
```

The configuration resource manager will at this time supply Topology Services on the new node with the latest cluster definition for the peer domain. This will extend 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, will be invited to go online.

For more information about the **startrpdomain** command, refer to the directions for creating a peer domain (the **startrpdomain** command is described in more detail in “Bringing the peer domain online” on page 29 of those directions). For complete syntax information on the **startrpnode**, **startrpdomain**, **lsrpnode**, or **lsrpdomain** commands, refer to their man pages in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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

In order to perform node maintenance or make application upgrades, you might want to take individual nodes of a peer domain offline or take an entire peer domain offline. You can:

- 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 to perform application upgrades, to perform maintenance on a node, or prior to removing the node from the peer domain (as described in “Removing a node from a peer domain” on page 38). Also, since a node may 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 may 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                 2.2.1.20
nodeB                   online                  2.2.1.20
nodeC                   online                  2.2.1.20
nodeD                   online                  2.2.1.20
nodeE                   offline                 2.2.1.20
```

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

```
stoprpnode nodeA
```

You can also take multiple nodes offline. For example:

```
stoprpnode nodeA nodeB
```

An RSCT subsystem (such as Topology Services or Group Services) may reject the **stoprpnode** command's 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's **-f** flag. For example:

```
stoprpnode -f nodeA
```

To later bring the node back online, use the **startrpnode** command as described in "Bringing the node online in the peer domain" on page 35. For complete syntax information on the **stoprpnode** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Taking a peer domain offline

In order to perform maintenance on a peer domain, you might wish to take it offline. 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 wish 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) may 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. For more information on the **startrpdomain** command, refer to "Bringing the peer domain online" on page 29. For complete syntax information on the **stoprpdomain** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 it is unknown why the node is remaining in the pending online state, you should capture a ctsnap prior to running the command.

For complete syntax information on the **forcerpoffline** command, refer to its man page in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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. You can:

- 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, refer to “Taking a peer domain node offline” on page 36.

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                2.2.1.20
nodeB                     online                 2.2.1.20
nodeC                     online                 2.2.1.20
nodeD                     online                 2.2.1.20
nodeE                     offline                2.2.1.20
```

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
```

For complete syntax information on the **rmrpnode** and **lsrpnode** commands, refer to their man pages in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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. You should 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, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Changing a peer domain's quorum type

As described in "What are quorum types?" on page 19, 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, refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

| ## 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.
- On the rare occasion when it might be necessary, manually configure communication groups in a peer domain.

"Understanding communication groups" describes the concepts you need to know before doing any of the these tasks.

| ## 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 by 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 will create 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 by the **addrpnode** command. When these added nodes are brought online by the **startrpnode** command, the configuration resource manager supplies the modified information to Topology Services which may modify existing heartbeat rings or create additional heartbeat rings.

The configuration resource manager's automatic creation of communication groups is based on subnet and intersubnet accessibility. For each communication group, the goal is to define a set of adapters (with no more than one adapter 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 adapters in the same subnet will be in the same communication group, unless one node has multiple adapters in the same subnet.
- adapters in different subnets that can communicate with each other may be in the same communication group if they have connectivity.

The configuration resource manager allows you to create your own communication groups and also change the adapter membership in an existing communication group. However, since the configuration resource manager will create the communication groups automatically, such manual configuration is neither necessary or advisable. **Manual configuration may be exercised, but only in unavoidable situations** (such as when a network configuration is more complex than our automatic communication group creation algorithm has anticipated and can handle). Manual configuration changes that do not conform to the above rules and restrictions may cause partitioning of the peer domain. For more information, refer to “Manually configuring communication groups” on page 44.

When the configuration resource manager automatically creates communication groups, it gives them default characteristics such 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 42.

Listing communication groups

The **lscomg** command lists information about the communication groups in a peer domain. It lists the:

- name of the communication group
- the sensitivity setting (the number of missed heartbeats that constitute a failure)
- the period setting (the number of seconds between heartbeats)
- the priority setting (the relative priority of the communication group)
- whether or not broadcast should be used if it is supported by the underlying media
- whether or not source routing should be used if it is supported by the underlying media
- the path to the Network Interface Module (NIM) that supports the adapter types in the communication group
- the NIM start parameters
- the name of the resource interface that refers to this communication group
- the peer domain node name of the resource interface that refers to this communication group
- the IP address of the resource interface that refers to this communication group
- the subnet mask of the resource interface that refers to this communication group
- the subnet of the resource interface that refers to this communication group

For example, to list general information about the peer domain *ApplDomain*, 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
ComG1  2            2        1         no          yes
NIMPath                      NIMParameters
/usr/sbin/rsct/bin/hats_nim  -l 5
```

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 the communication group *ComGrp*, enter:

```
lscomg ComGrp
```

To list interface resource information for a communication group, use the **-i** flag on the **lscomg** command.

```
lscomg -i ComGrp1
```

Output is similar to:

```
IName  IHostName    IIPAddr      ISubnetMask    ISubnet
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 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." For complete syntax information on the **lscomg** command, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying a communication group's characteristics

A communication group has a number of properties 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 41.

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 44 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, refer to its man page in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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. To determine what a communication group's sensitivity setting is, use the **lscomg** command as described in "Listing communication groups" on page 41. To modify a communication group's sensitivity setting, use the **chcomg** command with its **-s** flag. For 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
```

The sensitivity setting must be an integer greater than or equal to 2.

Modifying a communication group’s period setting

A communication group’s period setting refers to the number of seconds between Topology Service’s heartbeats. To determine what a communication group’s period setting is, use the **lscomg** command as described in “Listing communication groups” on page 41. To modify a communication group’s period setting, use the **chcomg** command with its **-p** flag. For 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
```

The period setting must be an integer greater than or equal to 1.

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. To determine what a communication group’s priority setting is, use the **lscomg** command as described in “Listing communication groups” on page 41. To modify a communication group’s priority setting, use the **chcomg** command with its **-t** flag. For 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). To determine what a communication group’s broadcast setting is, use the **lscomg** command as described in “Listing communication groups” on page 41. To modify a communication group’s broadcast setting so that broadcast operations are enabled, use the **chcomg** command with its **-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 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 **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 41.

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

"Modifying a communication group's characteristics" on page 42 describes how to use the **chcomg** command to modify a communication group's tunables (such as its sensitivity, period, and priority settings). You can also use the **chcomg** command to modify a communication group's network interface assignment. 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:

1. at most one adapter from each node can belong to a given communication group.
2. given the restriction in (1), all adapters in the same subnet will be in the same communication group.
3. given the restriction in (1), 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 = 1.2.3.4 and netmask = 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.

To modify 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.
- 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 41 to ascertain this.

For example, 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.

To create a communication group, use the **mkcomg** command. One of the key things you'll 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:

1. at most one adapter from each node can belong to a given communication group.

2. given the restriction in (1), all adapters in the same subnet will be in the same communication group.
3. given the restriction in (1), 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

For example, 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.

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.

For example, 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 41). To change any of the settings, you can use the **chcomg** command (as described in “Modifying a communication group’s characteristics” on page 42). For complete syntax information on the **mkcomg** command, refer to its man page in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 41. 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 44).

To remove a communication group, simply supply its name to the **rmcomg** command. For 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, refer to its man page in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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, refer to Chapter 8, “The Topology Services subsystem,” on page 235. For a complete discussion of Group Services, refer to Chapter 9, “The Group Services subsystem,” on page 253. 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 87).

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**).

Changing IP addresses in a peer domain

The configuration resource manager automatically monitors for configuration changes (such as IP address changes) in the 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. Since IP addresses are the critical path to a node, there are a couple of 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 9 outlines the rules for changing IP addresses in a peer domain.

Table 9. Changing IP addresses in a peer domain

If a node has...	**Then...**
Multiple IP addresses and you want to change only a subset of them	There are no restrictions to changing IP addresses.
Multiple IP addresses and you want to change all of them	You must not change all 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, you should wait about 1 minute and 30 seconds for the change to be reflected or until the command `lsrsrc IBM.NetworkInterface` reflects the change. 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`

Table 9. Changing IP addresses in a peer domain (continued)

If a node has...	Then...
A single IP address	This is the only access to the node. You should: 1. Remove the node from the peer domain (using the **rmrpnode** command, as described in "Removing a node from a peer domain" on page 38). 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 32).

Determining how your system responds to domain partitioning and subsystem daemon failure

In order 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, refer to "What is operational quorum?" on page 18.

There are various ways 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. Refer to Chapter 4, "Managing and monitoring resources using RMC and resource managers," on page 63 for more information.

You can:

- Determine the way critical resources are protected should a domain lose operation quorum or if the configuration manager, group services, or topology services daemons die or hang. This is done by setting the CritRsrcProtMethod attribute of the IBM.PeerNode class (or an individual IBM.PeerNode instance).
- Specify that the peer domain should 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 will not be protected.
- Set the active tie breaker that the configuration resource manager will use to resolve tie situations when two or more sub-domains containing exactly half the defined nodes are competing for operational quorum. In addition, you can modify a tie-breaker definition, define a new tie breaker, explicitly resolve a tie when the active tie-breaker type is ″Operator″.

For complete syntax information on the generic RMC commands (such as **lsrsrc** and **chrsrc**), refer to their man pages in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Setting the critical resource protection method for a peer domain or a node in a peer domain

When an RSCT peer domain is partitioned into two or more sub-domains, the configuration resource manager will determine which sub-domain has operational quorum and will survive, and which others should be dissolved. If the sub-domain 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 has lost quorum (as indicated by the PeerNode resource's CritRsrcActive dynamic attribute), 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 is essential, since certain applications require shared resource access. When a domain is partitioned, each sub-domain is unaware of any other sub-domain, and so multiple sub-domains may simultaneously access the shared resource and, in doing so, cause data corruption. A node's critical resource protection method is also needed if the configuration manager, group services, or topology services daemons die or hang.

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 will be 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 10 shows the possible settings for the CritRsrcProtMethod persistent attribute.

Table 10. CritRsrcProtMethod Settings

CritRsctProtMethod persistent attribute value	**Description**
1	Hard reset and reboot.
2	Halt system.
3	Sync, hard reset and reboot.
4	Sync, Halt system.
5	None.
6	Exit and restart RSCT subsystems.

For the IBM.PeerNode resource class, the default value of CritRsctProcMethod is 1 (hard reset and reboot). For the individual resource instances of IBM.PeerNode, the CritRsrcProtMethod persistent attribute can also have the value 0 which is the default and means that the resource instance inherits the value from the resource class.

To view or set the critical resource protection method for a peer domain or a node in the peer domain, use the standard RMC management commands **lsrsrc** and **chrsrc**.

For example, to list the current value of the CritRsrcProtMethod persistent attribute for each node in the domain, you would 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 peer domain's overall critical resource protection method. To list the domain-wide attributes, 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 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
```

For complete syntax information on the **lsrsrc** and **chrsrc** commands, refer to their man pages in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or in *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Overriding the configuration resource manager's operational quorum calculation to force operational quorum

When a peer domain is partitioned, the configuration manager will, by default, determine which sub-domain has operational quorum using the following calculation:

```
If (( 2*numNodesOnline ) > numNodesDefined )
            OpQuorumState = HasQuorum
If (( 2*numNodesOnline ) == numNodesDefined )
            OpQuorumState = PendingQuorum
            (until tie breaker 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). You should 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 11 shows the possible settings for the OpQuorumOverride persistent class attribute of the IBM.PeerNode resource class.

Table 11. OpQuorumOverride Settings

OpQuorumOverride 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, refer to their man pages in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Determining how the configuration resource manager will resolve tie situations when calculating operational quorum

When a peer domain is partitioned, the configuration resource manager must determine which sub-domain has operational quorum and so will survive, and which sub-domain will be dissolved. Often, this is simply a case of determining which of the sub-domains has more than half of the nodes. In the case of a tie in which the peer domain has been partitioned into two sub-domains 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 sub-domain 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 tie breaker

The OpQuorumTieBreaker persistent class attribute of the IBM.PeerNode class indicates the active tie breaker for the peer domain. There may be a number of tie breakers (IBM.TieBreaker resources) defined for the peer domain, but only one may be active at a time. The configuration resource manager will use this active tie breaker after domain partitioning if there are multiple sub-domains with the same number of nodes to determine which sub-domain will have operational quorum. There are two predefined tie-breaker resources, and you can also define your own as described in “Defining a new tie breaker” on page 55. Table 12 on page 53 describes the two predefined tie breakers: Operator and Fail.

Table 12. Predefined tie breakers (IBM.TieBreaker resources)

Tie breaker	Description
Operator	The system administrator resolves the tie by invoking the ResolveOpQuorumTie action of the IBM.PeerDomain resource class. Until the administrator explicitly breaks the tie, neither domain will have operational quorum. The OpQuorumState dynamic attribute of the PeerDomain resource will be 1 (PendingQuorum) until the administer invokes the ResolveOpQuorumTie action. For more information, refer to "Explicitly resolving a tie when the active tie-breaker type is ″Operator″" on page 60.
Fail	A pseudo tie breaker in that it does not actually resolve the tie situation. Neither sub-domain will have operational quorum. The OpQuorumState dynamic attribute of each PeerDomain resource will be 2 (NoQuorum). If critical resources are active on a domain that has lost quorum (as indicated by the PeerDomain resource's CritRsrcActive dynamic attribute), 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. See "Setting the critical resource protection method for a peer domain or a node in a peer domain" on page 49 for more information on critical resource protection methods.

To view or set the active tie breaker (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 tie breaker, you would 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 us that the current active tie breaker is ″Fail″. To list the names of all of the available tie breaker resources, you would 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"
```

To make the ″Operator″ tie breaker the active tie breaker, you would use the **chrsrc** command with its **-c** flag.

```
chrsrc -c IBM.PeerNode OpQuorumTieBreaker="Operator"
```

If you set the active tie breaker to ″Operator″, then, should a tie situation occur, you will need to manually resolve the tie by invoking the ResolveOpQuorumTie action of the IBM.PeerDomain resource class. Refer to "Explicitly resolving a tie when the active tie-breaker type is ″Operator″" on page 60 for more information.

For complete syntax information on the **lsrsrc** and **chrsrc** commands, refer to their man pages in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying a tie-breaker definition

A tie breaker (IBM.TieBreaker resource) has a number of persistent resource attributes that you can set to configure the tie breaker's behavior. 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 tie breakers, you would 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            = {}
```

To limit the output of the **lsrsrc** command to display the persistent attribute values for only a particular tie breaker resource, you could use the **-s** flag and a selection string that identifies the particular tie breaker 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 on any of these persistent attributes, use the **lsrsrcdef** command (as described in "Displaying attribute definition information for a resource or a resource class" on page 87). To change the persistent attributes of a tie breaker, the tie breaker must not be the active tie breaker. The OpQuorumTieBreaker persistent class attribute of the IBM.PeerNode class identifies the active tie breaker. If you are not sure if the tie breaker you want to modify is the active tie breaker, 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"
```

For instructions on the setting the OpQuorumTieBreaker persistent class attribute, refer to “Setting the active tie breaker” on page 52. As long as the tie breaker is not the active tie breaker, you can modify its persistent resource attributes using the **chrsrc** command. To identify a particular tie breaker, you will need to use the **chrsrc** command’s **-s** flag followed by a selection string that identifies the tie breaker resource. For example:

```
chrsrc -s"Name=='Operator'" IBM.TieBreaker ReleaseRetryPeriod=30
```

For complete syntax information on the **lsrsrc** and **chrsrc** commands, refer to their man pages in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Defining a new tie breaker

In addition to the predefined tie breakers, you can also create your own by defining a new IBM.TieBreaker resource using the standard RMC management command **mkrsrc**.

Attention: When defining tie breaker resources, be aware that the disk on which IBM.Tiebreaker resources are stored should not also be used to store file systems.

When defining a tie breaker, you need to first determine which persistent attributes are required when defining an IBM.TieBreaker resource. This information can be returned by issuing the **mkrsrc** command with its **-e** command-line flag. The **-e** flag causes the **mkrsrc** command to display two examples of suitable command-line input when defining a given 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 on any of the attributes of an IBM.TieBreaker resource, use the **lsrsrcdef** command (as described in “Displaying attribute definition information for a resource or a resource class” on page 87). Here, however, we will focus only on the two that are required for defining an IBM.TieBreaker resource — the Type and Name attributes.

- The Type attribute is the name of one of the available tie-breaker types. The available tie breaker types will depend on your operating system and machine architecture. Possible types are:

  Operator
  : This type of tie breaker asks for a decision from the system operator or administrator. The operator executes his decision by invoking the ResolveOpQuorumTie action as described in “Explicitly resolving a tie when the active tie-breaker type is ″Operator″” on page 60.

  Fail
  : This pseudo tie breaker type always fails to reserve the tie breaker.

  ECKD™
  : This tie breaker type is specific to Linux for zSeries®. This tie breaker type assumes that an ECKD-DASD is shared by all nodes of the cluster. Tie breaker reservation is done by the ECKD reserve command. If creating a tie breaker of this type, you need to set the DeviceInfo

persistent resource attribute to indicate the ECKD device number. See “Creating an ECKD tie breaker” for more information.

SCSI This tie breaker type is specific to Linux for xSeries®. This tie breaker type assumes that an SCSI-disk is shared by one or more nodes of the peer domain. Tie breaker reservation is done by the SCSI reserve or persistent reserve command. If creating a tie breaker of this type, you need to set the DeviceInfo persistent resource attribute to identify the SCSI device. See “Creating an SCSI tie breaker” on page 57 for more information.

DISK This tie breaker type is specific to AIX. This tie breaker type enables you to specify a SCSI or SCSI-like physical disk using an AIX device name, and assumes that the SCSI disk is shared by one or more nodes of the peer domain. Tie breaker reservation is done by the SCSI reserve or persistent reserve command. If creating a tie breaker of this type, you need to set the DeviceInfo persistent resource attribute to identify the physical disk. Only SCSI and SCSI-like physical disks are supported. Physical disks attached via Fiber Channel, iSCSI and Serial Storage Architecture Connections are suitable.

EXEC A script provided by an RSCT exploiter product resolves the tie situation.

The tie breaker 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 tie breaker types, you would 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",""]}
```

If the **lsrsrc** command example shown above is issued on a Linux zSeries machine, the output would show ECKD as one of the available types. If issued on a Linux xSeries machine, the output would show SCSI as an available type. If issued on an AIX machine, the output would show DISK as an available type.

- The Name attribute is simply a null-terminated string you will use to identify this tie breaker. It is the value you will use when setting the OpQuorumTieBreaker persistent class attribute of the IBM.PeerNode resource class to activate the tie breaker. See “Setting the active tie breaker” on page 52 for more information.

Once you understand the values you want to assign to the persistent attributes that are required for define (and any attributes that are optional for define that you want to specify), you define the IBM.TieBreaker resource using the **mkrsrc** command. For example:

```
mkrsrc IBM.TieBreaker Name=OpQuorumTieBreaker Type=Operator
```

For complete syntax information on the **lsrsrcdef**, **lsrsrc** and **mkrsrc** commands, refer to their man pages in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Creating an ECKD tie breaker: The ECKD tie-breaker type is specific to Linux on zSeries. If you want to create an ECKD tie breaker object, you need to set the DeviceInfo persistent resource attribute to indicate the ECKD device number. This type of tie breaker 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 tie breaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation is retried.

Attention: When defining tie breaker 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
```

Creating an SCSI tie breaker: The SCSI tie-breaker type is specific to Linux on xSeries. If you want to create a SCSI tie breaker object, you need to specify the SCSI device 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.

This type of tie breaker 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 tie breaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation is retried.

Attention: When defining tie breaker resources, be aware that the disk on which IBM.Tiebreaker resources are stored should not also be used to store file systems.

To obtain the identifiers for a SCSI device, enter:

```
cat /proc/scsi/scsi
```

Output similar to the following is displayed:

```
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM      Model: DRVS18D         Rev: 0380
  Type:   Direct-Access                   ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 01 Lun: 00
  Vendor: IBM      Model: DRVS18D         Rev: 0380
  Type:   Direct-Access                   ANSI SCSI revision: 03
```

```
Host: scsi0 Channel: 00 Id: 15 Lun: 00
  Vendor: IBM      Model: 2104-TL1         Rev: BP18
  Type:   Enclosure                        ANSI SCSI revision: 03
```

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, you identify the SCSI device using the DeviceInfo persistent resource attribute. For example:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI DeviceInfo="ID=4 LUN=0 HOST=0,CHAN=0" \
       HeartbeatPeriod=30
```

Because the SCSI configuration can be different between nodes (even if the target device is the same), you may need to reflect differences between nodes using the NodeInfo persistent resource attribute. For example, say a 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 will only have to use the NodeInfo attribute to specify the nodes that are different.

| If the RDAC driver is installed, a SCSI tie breaker can also be specified using the
| WWN (world-wide number). To obtain the WWN for each disk, enter:

```
cat /proc/scsi/mpp/* | grep "^[ ]*Lun"
```

| The output will be similar to the following:

```
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 the above display, each disk is associated with a WWN value and, on the following line, a LunObjectk value. For instance, the following portion of the display shows the WWN and LunObject values associated with Lun #6:

```
    Lun #6 - WWN:600a0b80000cf57b0000000d406ec56e
       LunObject: h4c0t0l6                              CurrentOwningPath: A
```

The format of the LunObject value (for instance, `h4c0t0l6`, for Lun #6 in the above display) is: **h***host_number***c***channel_number***t***scsi_id***l***lun*

To create a tie breaker object using a WWN, issue the **mkrsrc** command as in the following example:

```
mkrsrc IBM.TieBreaker Name=scsi Type=SCSI
       DeviceInfo="WWN=600a0b80000cf57b0000000d406ec56e" HeartbeatPeriod=30
```

Creating a DISK tie breaker: The DISK tie-breaker type is specific to AIX. If you want to create a DISK tie breaker object, you need to set the DeviceInfo persistent resource attribute to indicate the AIX device name. The AIX device name must specify a SCSI or SCSI-like physical disk that is shared by all nodes of the peer domain. Physical disks attached via Fiber Channel, iSCSI, and Serial Storage Architecture may serve as a DISK tie breaker. However, IDE hard disks do not support the SCSI protocol and cannot serve as a DISK tie-breaker. Logical volumes also cannot serve as a DISK tie breaker.

This type of tie breaker 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 tie breaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation is retried.

Attention: When defining tie breaker 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 tie breaker, 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, 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.

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 tie breaker, as follows:

```
mkrsrc IBM.TieBreaker Name=disktb Type=DISK DeviceInfo="PVID=000000371e5766b8" \
      HeartbeatPeriod=30
```

Creating an EXEC tie breaker: You can only create a tie breaker 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 tie breaker object using the **mkrsrc** command. For the EXEC tie breaker type, the DeviceInfo attribute should specify the path to the script or executable and any program arguments. For example, to create the tie breaker named **MyTB** when the provided executable is **/usr/sbin/rsct/bin/tiebreaker**, you would enter:

```
mkrsrc -c IBM.TieBreaker Type="EXEC" Name="MyTB" \
      DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/tiebreaker myArg=123'
```

Explicitly resolving a tie when the active tie-breaker type is ″Operator″

When the active tie breaker is the predefined tie breaker ″Operator″ or a tie breaker whose persistent attribute Type is ″Operator″, then the configuration resource manager will not automatically resolve tie situations. If domain partitioning occurs with a sub-domain containing exactly half the defined nodes (or if exactly half of the domain's defined nodes become inaccessible), the configuration manager will set the OpQuorumState dynamic attribute of the PeerDomain resource to 1 (PendingQuorum). Operational quorum will not be 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 sub-domain. The single input parameter to this action is an integer that indicates whether the sub-domain in which the action is invoked is denied (0) or granted (1) or ownership of the tie breaker.

When explicitly resolving a tie between sub-domains, you should, in order to avoid corruption of shared data, first deny ownership of the tie breaker to the appropriate sub-domain. Once you have denied ownership of the tie breaker to the appropriate sub-domain, you can safely grant ownership of the tie breaker to the sub-domain that you want to have operational quorum.

To deny ownership of the ″Operator″ tie breaker to a sub-domain, invoke the following action on a node of that sub-domain.

```
runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=0
```

Denying ownership of the tie breaker to a sub-domain will cause the configuration manager to to set the OpQuorumState dynamic attribute of the PeerDomain resource to 2 (NoQuorum). The sub-domain will lose quorum, which may in turn cause the critical resource protection method to be invoked on any nodes that have critical resources active. See “Setting the critical resource protection method for a peer domain or a node in a peer domain” on page 49 for more information.

To grant ownership of the ″Operator″ tie breaker to a sub-domain, invoke the following action on a node of that sub-domain.

```
runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=1
```

Granting ownership of the tie breaker to a sub-domain will cause the configuration manager to set the OpQuorumState dynamic attribute of the PeerDomain resource to 0 (HasQuorum). The sub-domain will have operational quorum and so will become the peer domain.

Chapter 4. Managing and monitoring resources using RMC and resource managers

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 108.

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. 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.

Understanding RMC and resource managers

Before performing resource management tasks, you should gain an understanding of 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 88 and "Advanced resource monitoring" on page 105).
- How RMC implements authorization (described in more detail later in "Managing user access to resources using RMC ACL files" on page 80).
- Differences between using RMC on a single node versus a cluster.

What is RMC?

The Resource Monitoring and Control (RMC) 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.

What is 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.

What is 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).

What are 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 is a fairly simply task, this same 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, the number of users currently logged on, and so on.

A dynamic attribute is further classified by its *variable type*, which will be one of the following:

Variable type	**Description**
Counter	This variable type represents a rate.

Quantity	This variable type represents a value that fluctuates over time, and typically represents a level.
State	This variable type represents a value that fluctuates over time in which every change is important.
Quantum	Signifies a change, but has no value associated with it.

Dynamic attributes are useful in monitoring your system for conditions of interest. As described in “Basic resource monitoring” on page 88 and “Advanced resource monitoring” on page 105, 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 an attribute. 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.

What is 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, refer to their online man pages. Detailed information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

What is 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 108) 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 88 and “Advanced resource monitoring” on page 105) 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 Chapter 3, “Creating and administering an RSCT peer domain,” on page 17) 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 three startup modes are:

auto-start
The resource manager is started when the RMC subsystem is started.

on-line auto-start
: The resource manager is started when the node becomes online in a peer domain.

on-demand
: The resource manager is started when one of its resources is first referenced.

The startup mode for each RSCT resource manager is indicated in Table 13 in "What resource managers are provided with RSCT?."

What resource managers are provided with RSCT?

Table 13 describes the resource managers that are provided as part of RSCT. Together with the RMC subsystem, they provide the administrative and monitoring capabilities of RSCT. Keep in mind that additional resource managers are provided by certain cluster licensed program products (such as CSM, which contains the Domain Management resource manager).

Table 13. Resource Managers Provided with RSCT

Resource manager	Description	Startup mode
Audit log resource manager	Provides a system-wide facility for recording information about the system's operation. It is use by subsystem components to log information about their actions, errors, and so on. In particular, the Event Response resource manager, which contains the resource monitoring functionality, uses the audit log resource manager to log information about condition events occurring, what responses were taken, and so on. A command-line interface to the audit log resource manager enables you to list and remove records from and audit log. For more information on the audit log resource manager's commands, refer to "Using the audit log to track monitoring activity" on page 97. Complete syntax information on the commands is provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*. For more information, refer to "Resource classes defined by the audit log resource manager" on page 68.	on-demand
CIM resource manager	Enables you to use RMC to query system configuration through Common Information Model (CIM) classes. The CIM resource manager provides a command (**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. For more information, refer to "Querying and monitoring CIM properties and associations" on page 129. Complete syntax information on the **mkcimreg** command is provided in the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.	on-demand
Configuration resource manager	Provides the ability, through its command-line interface, to create and administer a peer domain (a cluster of nodes configured for high availability). For more information on the configuration resource manager's commands, refer to Chapter 3, "Creating and administering an RSCT peer domain," on page 17. Complete syntax information on the commands is provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*. For more information, refer to "Resource classes defined by the configuration resource manager" on page 69.	on-demand (if offline). Otherwise on-line auto-start
Event response resource manager	Provides resource monitoring — the ability to take actions 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. For more information on the event response resource manager's commands, refer to "Basic resource monitoring" on page 88 and "Advanced resource monitoring" on page 105. Complete syntax information on the commands is provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*. For more information, refer to "Resource classes defined by the event response resource manager" on page 70.	auto-start
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. For more information, refer to "Resource classes defined by the file system resource manager" on page 70.	on-demand

Table 13. Resource Managers Provided with RSCT (continued)

Resource manager	Description	Startup mode
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 using the event response resource manager. For more information, refer to “Resource classes defined by the host resource manager” on page 70.	on-demand
Least-privilege resource manager	Controls access to root commands or scripts, and local or remote execution of those commands or scripts on AIX or Linux. The least-privilege (LP) resource manager provides a resource class and a command-line interface that allow you to define, manage, and monitor root commands and scripts as LP resources. For more information about the LP resource manager, refer to “Resource classes defined by the least-privilege resource manager” on page 71 and Chapter 5, “Controlling access to root commands and scripts,” on page 153.	on-demand
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 have defined. The sensor resource manager provides a command-line interface for creating, changing, listing, and removing sensors. For more information on the sensor resource manager’s commands, refer to “Creating event sensor commands for monitoring” on page 127. Complete syntax information on the commands is provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*. For more information, refer to “Resource classes defined by the sensor resource manager” on page 71.	on-demand
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 manger 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 manger with a global view of the storage resources. For more information, see Chapter 6, “Understanding and administering the Storage resource manager,” on page 183.	on demand

Resource classes defined by the audit log resource manager: Table 14 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 97) and **rmaudrec** command (as described in “Deleting entries from the audit log using the rmaudrec command” on page 102). 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 103).

Table 14. 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, refer to “Listing resource information” on page 83.

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, refer to “Querying and monitoring CIM properties and associations” on page 129.

Resource classes defined by the configuration resource manager: Table 15 describes the resource classes defined by the configuration resource manager. In general, you will not need to manipulate resources of these classes directly. Instead you would use the configuration resource manager commands described in Chapter 3, “Creating and administering an RSCT peer domain,” on page 17. However, you may need to:

- modify attributes of the RSCTParameters class (as described in “Modifying Topology Services and Group Services parameters” on page 47.
- modify the CritRsrcProtMethod attribute of the IBM.PeerNode class or and resource instance of the IBM.PeerNode class (as described in “Setting the critical resource protection method for a peer domain or a node in a peer domain” on page 49).
- modify the OpQuorumOverride attribute of the IBM.PeerNode class (as described in “Overriding the configuration resource manager’s operational quorum calculation to force operational quorum” on page 51).
- modify the OpQuorumTieBreaker attribute of the IBM.PeerNode class (as described in “Setting the active tie breaker” on page 52).

Table 15. 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 ″heartbeating″) will be performed.
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.
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.

Table 15. Resource classes defined by the configuration resource manager (continued)

Resource class	Description
IBM.TieBreaker	Each resource of this class represents a tie-breaker. A tie-breaker is used, when domain partitioning results in two sub-domains each containing exactly half the nodes, to determine which sub-domain has operational quorum.

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, refer to "Listing resource information" on page 83.

Resource classes defined by the event response resource manager: Table 16 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 88 and "Advanced resource monitoring" on page 105.

Table 16. 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.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, refer to "Listing resource information" on page 83.

Resource classes defined by the file system resource manager: Table 17 describes the resource class 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. Monitoring is described in "Basic resource monitoring" on page 88 and "Advanced resource monitoring" on page 105.

Table 17. 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, refer to "Listing resource information" on page 83.

Resource classes defined by the host resource manager: Table 18 on page 71 describes the 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. Monitoring is described in "Basic resource monitoring" on page 88 and "Advanced resource monitoring" on page 105.

Table 18. Resource classes defined by the host resource manager

Resource class	Description
IBM.ATMDevice	Each resource of this class represents an ATM adapter installed on the host.
IBM.EthernetDevice	Each resource of this class represents an Ethernet adapter installed on the host.
IBM.FDDIDevice	Each resource of this class represents a FDDI adapter installed on the host
IBM.Host	Each resource of this class represents a host machine that is running a single copy of an operating system.
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.
IBM.PhysicalVolume	Each resource of this class represents a physical volume.
IBM.Processor	Each resource of this class represents a processor.
IBM.Program	Each resource of this class represents a program that is running on the host.
IBM.TokenRingDevice	Each resource of this class represents a token-ring device installed on the host.

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, refer to “Listing resource information” on page 83.

Resource classes defined by the least-privilege resource manager: Table 19 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 Chapter 5, “Controlling access to root commands and scripts,” on page 153.

Table 19. 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, refer to “Listing resource information” on page 83.

Resource classes defined by the sensor resource manager: Table 20 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 127.

Table 20. 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, refer to “Listing resource information” on page 83.

Resource classes defined by the storage resource manager: Table 21 describes the 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 21. 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 or AIX logical volume. These attributes are a subset of the entries in the IBM.FileSystem class of the File System resource manager.
IBM.Disk	This resource class externalizes the attributes of SCSI disks on Linux and physical volumes which are sole members of a volume group on AIX.
IBM.LogicalVolume	This resource class externalizes the attributes of logical volumes configured on AIX volume groups. The logical volume entities are not harvested on Linux nodes, but AIX logical volume resources may still be managed from Linux nodes in a heterogeneous cluster.
IBM.Partition	On Linux nodes only, this resource class externalizes the attributes of any configured partitions on a disk device or any logical volumes that exist on resources of the IBM.Disk class. Not available on Linux nodes.
IBM.VolumeGroup	This resource class externalizes the attributes of volume groups comprised of only one physical volume on AIX nodes. Volume group entities are not harvested on Linux nodes, but AIX volume group resources may still be managed from Linux nodes in a heterogeneous cluster.

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, refer to “Listing resource information” on page 83. For more information on the Storage resource manager, see Chapter 6, “Understanding and administering the Storage resource manager,” on page 183.

How do 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, refer to “Listing resource information” on page 83.

For complete syntax and reference information on the generic RMC commands refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*

How do 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.

What is 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*).

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 on page 75 illustrates the cycle of event expression/rearm event expression evaluation.

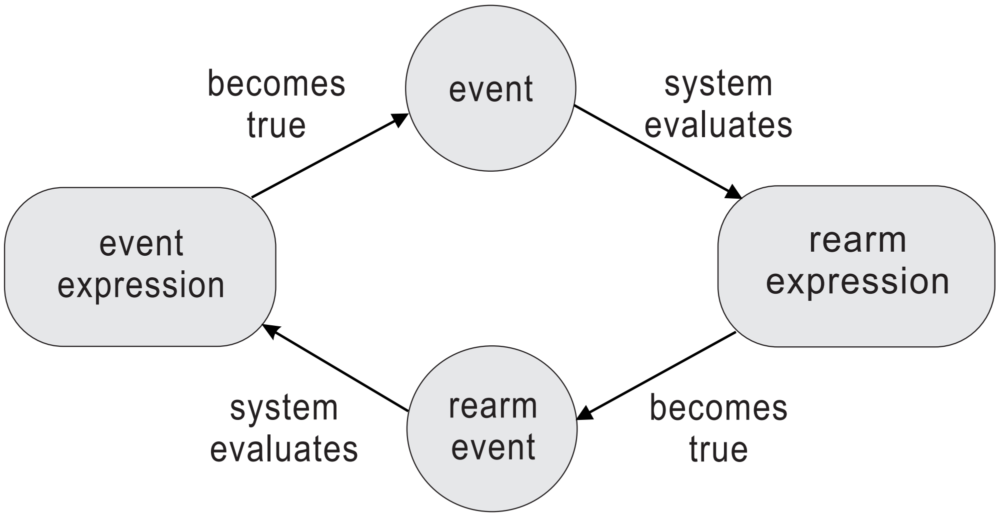

Figure 1. Cycle of event expression and rearm event expression evaluation for a condition

The monitoring scope of a condition: Another important feature of a condition is its *monitoring scope*. 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 since 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 89.

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 108. 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, refer to “Creating event sensor commands for monitoring” on page 127.

What is 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 simply 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.

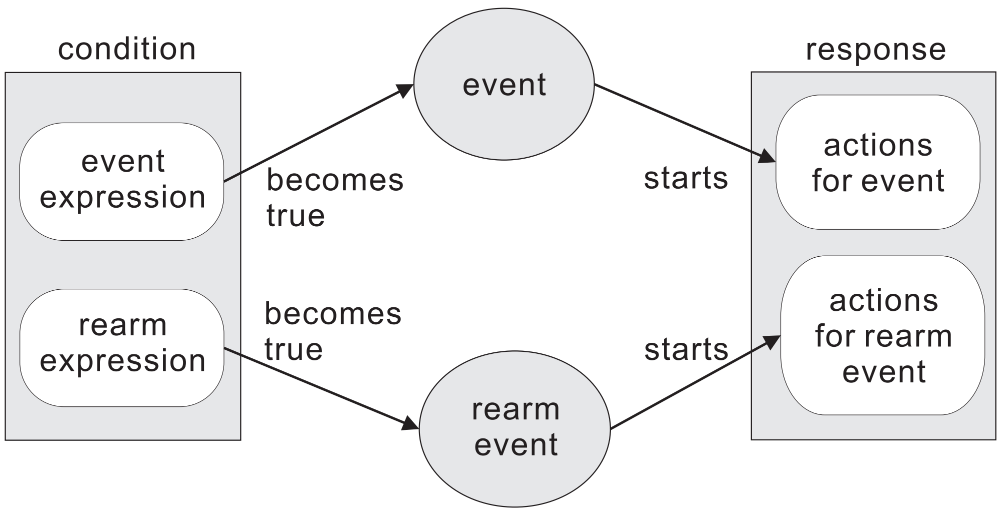

Figure 2. A response with separate actions for a condition's event and rearm event

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, say 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: You can 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 119), you can add a single action to the response. You can then use the **chresponse** command (as described in "Modifying a response" on page 125) 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 22 on page 77 describes the predefined responses that RSCT provides.

Table 22. 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.	8AM-5PM, 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
E-mail root off-shift	E-mail root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an e-mail to root when an event or a rearm event occurs.	5PM-12PM, Monday to Friday 12AM-8AM, Monday to Friday All day, Saturday and Sunday
E-mail root anytime	E-mail root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an e-mail 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
	E-mail root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an e-mail to root when an event or a rearm event occurs.	8AM-5PM, 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
	E-mail root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an e-mail 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
	E-mail root	Uses the predefined action script **/usr/sbin/rsct/bin/notifyevent** to send an e-mail to root when an event or a rearm event occurs.	All day, everyday
	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

What is 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. For more information refer to “Creating a condition/response association” on page 93 and “Starting condition monitoring” on page 94.

What should I monitor?

To get an idea of what you can monitor, take a look at the predefined conditions that RSCT provides. You can list the predefined conditions using the **lscondition** command (described in “Listing conditions” on page 89).

You can also create a condition based on any attribute of a resource class. Since persistent attributes are generally unchanging, 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 86) and the **lsrsrcdef** command (described in “Displaying attribute definition information for a resource or a resource class” on page 87).

Keep in mind that additional resource managers are provided by certain cluster licensed program products such as Cluster Systems Management (CSM), which provides the Domain Management Resource Manager. These additional resource managers may have resource classes with their own predefined conditions and their own attributes. Refer to the documentation for these licensed program products for details on any predefined conditions or attributes they provide.

One thing we can recommend that you monitor is the size of the **/var** file system. We recommend you do this because many RSCT subsystems make extensive use of this file system. To monitor the **/var** file system, you can use the predefined condition `/var space used` provided by the File System Resource Manager. If you are a CSM customer, you can also use the predefined condition `AnyNodeVarSpaceUsed` provided by the Domain Management Server Resource Manager. The Domain Management Server Resource Manager is only provided as part of CSM. The `AnyNodeVarSpaceUsed` condition monitors the **/var** file system on all nodes of the management domain.

How does RMC implement 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 80.

How do I determine the target nodes for a command?

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. (For more information on domains, refer to “What are management domains and peer domains?” on page 1.) It is important for you to understand that you can execute RMC and resource manager commands on a single machine, all the nodes of a peer domain, or all the nodes of a management domain. Some commands enable you to refine this even further, allowing you to specify a subset of nodes in the peer domain or management domain. When working in a cluster, you can also, from a local node, issue commands to be executed on another node.

There are two environment variables that, together with various command flags, determine the node(s) that will be affected by the RMC and resource manager commands you enter. These environment variables, CT_CONTACT and CT_MANAGEMENT_SCOPE, are described in Table 23.

Table 23. Environment variables that determine target nodes for a command

This environment variable...	Does this...
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 are: **0** the local scope. (This is either the local machine or the machine indicated by the CT_CONTACT environment variable). **1** the local scope. (This is either the local machine or the machine indicated by the CT_CONTACT environment variable). **2** the peer domain scope. (This is either the peer domain in which the local machine is online, or the peer domain in which the machine 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, and the ones that do may have command-line flags you can use to override the environment variable setting or otherwise determine how the command uses the specified values.

Note for CSM Users:

When the scope is set to the management domain scope (either through the CT_MANAGEMENT_SCOPE environment variable or through command line options), RMC commands issued from the management server will return information for managed nodes. Some of these nodes may also be in peer domains within the management domain.

Certain RMC class operations return information about a node's peer domain. When performing these operations in a management domain scope, some nodes might not be in a peer domain. In these cases, the peer domain field will simply provide the local host name. When a local host name is provided instead of a peer domain name, the name will appear in angle brackets (for example: <local_host_name>).

The *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference* 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 Node(s):

When this documentation discusses a command, it focuses on the command's basic function (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 Node(s)**″ note like this one.

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 83.

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, refer to Chapter 5, "Controlling access to root commands and scripts," on page 153.

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:

```
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 the forms shown in Table 24.

Table 24. 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 Chapter 7, "Understanding and administering cluster security services," on page 205). 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 forms.	
	This host user identifier format...	**Specifies...**
	user_name@host_identifier	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	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.
	*	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 229 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 25.

Table 25. 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 26.

Table 26. The optional permissions portion of the stanza line

Specifying this permission...	Indicates that the specified user(s) at the specified host(s) have...	
r	Read permission. This allows the user(s) 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:	
	Specifying this...	**Indicates that the specified user(s) at the specified host(s) have...**
	q	Query permission. This allows the user to query persistent or dynamic attributes.
	l	List permission. This allows the user to list resources.
	e	Event permission. This allows the user to register, query, and unregister events.
	v	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:	
	Specifying this...	**Indicates that the specified user(s) at the specified host(s) have...**
	d	Define permission. This allows the user to define and undefine resources.
	c	Refresh permission. This allows the user to refresh resource configuration.
	s	Set permission. This allows the user to set attributes.
	o	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.

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.

```
lsrsrc
```

Output will be 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"
"IBM.Sensor"
"IBM.PeerDomain"
"IBM.PeerNode"
"IBM.RSCTParameters"
"IBM.NetworkInterface"
"IBM.CommunicationGroup"
"IBM.HostPublic"
"IBM.TieBreaker"
"IBM.LPCommands"
"IBM.AgFileSystem"
"IBM.Disk"
"IBM.Partition"
"IBM.VolumeGroup"
"IBM.LogicalVolume"
```

To return detailed information on any of the resource classes, refer to the instructions in "Displaying attribute value information for a resource or a resource class" on page 86 and "Displaying attribute definition information for a resource or a resource class" on page 87.

Table 27 lists the resource classes defined by the RSCT resource managers.

Table 27. 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 72
IBM.Association	Event Response resource manager	"Resource classes defined by the event response resource manager" on page 70
IBM.ATMDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.AuditLog	Audit Log resource manager	"Resource classes defined by the audit log resource manager" on page 68
IBM.AuditLogTemplate	Audit Log resource manager	"Resource classes defined by the audit log resource manager" on page 68
IBM.CommunicationGroup	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 69
IBM.Condition	Event Response resource manager	"Resource classes defined by the event response resource manager" on page 70
IBM.Disk	Storage resource manager	"Resource classes defined by the storage resource manager" on page 72
IBM.EthernetDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.EventResponse	Event Response resource manager	"Resource classes defined by the event response resource manager" on page 70
IBM.FDDIDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.FileSystem	File System resource manager	"Resource classes defined by the file system resource manager" on page 70\
IBM.Host	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.HostPublic	Host resource manager	"Resource classes defined by the host resource manager" on page 70

Table 27. 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 72
IBM.LPCommands	Least-privilege resource manager	"Resource classes defined by the least-privilege resource manager" on page 71
IBM.NetworkInterface	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 69
IBM.PagingDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.Partition	Storage resource manager	"Resource classes defined by the storage resource manager" on page 72
IBM.PeerDomain	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 69
IBM.PeerNode	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 69
IBM.PhysicalVolume	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.Processor	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.Program	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.RSCTParameters	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 69
IBM.Sensor	Sensor resource manager	"Resource classes defined by the sensor resource manager" on page 71
IBM.TieBreaker	Configuration resource manager	"Resource classes defined by the configuration resource manager" on page 69
IBM.TokenRingDevice	Host resource manager	"Resource classes defined by the host resource manager" on page 70
IBM.VolumeGroup	Storage resource manager	"Resource classes defined by the storage resource manager" on page 72

In addition to the resource classes listed in Table 27 on page 84, keep in mind that:

- Any resource classes registered using the CIM resource manager (described in "Querying and monitoring CIM properties and associations" on page 129) may 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 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.
- Additional resource managers are provided by cluster licensed program products that exploit the RSCT technology (such as CSM, which provides the Domain Management resource manager). If you have any RSCT exploiter products installed on your system or cluster, the preceding command output may show additional resource classes that are defined by resource managers of the RSCT exploiter(s). For information on any additional resource classes not described in the following table, refer to the appropriate RSCT exploiter documentation.

For complete syntax information on the **lsrsrc** and **lsrsrcdef** commands, refer their online man pages. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 28, 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 28. 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 will return a list of the attributes requested. The attribute name and value will be listed.

Example: To list the persistent and dynamic attributes of for resources of the IBM.Host class, enter:

```
lsrsrc -A b IBM.Host
```

Output will be similar to the following:

```
Resource Persistent and Dynamic Attributes for IBM.Host
resource 1:
        Name                = "jbrady.ibm.com"
        NumProcessors       = 4
        RealMemSize         = 1073676288
        OSName              = "AIX"
        KernelVersion       = "5.3"
        DistributionName    = "IBM"
        DistributionVersion = "5300-04-00"
        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
⋮
```

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" on page 87.

Targeting Node(s):
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 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 **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 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, refer to the **lsrsrc** command man page and "How do I determine the target nodes for a command?" on page 78.

For complete syntax information on the **lsrsrc** command, refer its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 29, enable you to display attribute definition information

- for the resource class or resource instances of the class
- for persistent attributes or dynamic attributes

Table 29. 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 29 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, simply include the attribute name(s) on the **lsrsrcdef** command line.

Example: `lsrsrcdef -e IBM.Host KernelVersion`

Targeting Node(s):
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, refer to the **lsrsrcdef** command man page and "How do I determine the target nodes for a command?" on page 78.

For complete syntax information on the **lsrsrcdef** command, refer its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Basic resource monitoring

You can use Event Response Resource Manager commands to monitor your system of cluster domains. As described in "How do RMC and the resource managers enable you to monitor resources?" on page 73, 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 and responses rather than using the predefined ones provided by the various resource managers, refer to "Advanced resource monitoring" on page 105.

For detailed syntax information on any the commands discussed here, refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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.

Example: Enter the following at the command prompt:

```
lscondition
```

Results will be similar to the following:

```
|  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 108). 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"
```

Results will be similar to the following:

```
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 node(s):

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, refer to the **lscondition** command man page and "How do I determine the target nodes for a command?" on page 78.

For detailed syntax information on the **lscondition** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Listing responses

For a list of all available responses, enter the **lsresponse** command.

Example: Enter the following at the command prompt:

```
lsresponse
```

Results will be similar to the following:

```
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 119).

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 node(s):

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, refer to the **lsresponse** command man page and "How do I determine the target nodes for a command?" on page 78.

For detailed syntax information on the **lsresponse** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Listing condition/response associations

As described in "Listing conditions" on page 89 and "Listing responses" on page 90, many predefined conditions and responses are provided by the various resource managers on your system. What's more, you can create your own conditions and responses as described in "Advanced resource monitoring" on page 105. Before you can monitor a 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 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 with responses (as described in “Creating a condition/response association” on page 93), entering the **lscondresp** command will show the associations.

Example: The **lscondresp** output will be similar to the following:

```
Condition                 Response                      Node      State
"FileSystem space used"   "Broadcast event on-shift"    "nodeA"   "Active"
"FileSystem space used    "E-mail root anytime"         "nodeA"   "Not Active"
"Page in Rate"            "Log event anytime"           "nodeA"   "Active"
```

If you want to list the condition/response associations for a single condition, supply the name of the 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"
```

Output would be similar to the following:

```
Condition                 Response                      Node      State
"FileSystem space used"   "Broadcast event on-shift"    "nodeA"   "Active"
"FileSystem space used    "E-mail root anytime"         "nodeA"   "Not Active"
```

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
"FileSystem space used"   "Broadcast event on-shift"    "nodeA"   "Active"
```

Targeting node(s):

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, refer to the **lscondresp** command man page and “How do I determine the target nodes for a command?” on page 78.

For detailed syntax information on the **lscondresp** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster*

Technology for AIX 5L: Technical Reference or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Creating a condition/response association

Before you can 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. Many predefined conditions and responses are provided by the various resource managers on your system. What's more, you can create your own conditions and responses as described in "Advanced resource monitoring" on page 105.

| "Listing conditions, responses, and condition/response associations" on page 89
| 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 94.)

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 node(s):
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, refer to the **mkcondresp** command man page and "How do I determine the target nodes for a command?" on page 78.

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 91).

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.”

To prevent user modification or removal of a condition/response link, you can lock it (as described in “Locking and unlocking conditions, responses, and condition/response links” on page 140).

For detailed syntax information on the **mkcondresp** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and in *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Starting condition monitoring

The **startcondresp** command starts monitoring a 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 93), you can simply specify the name of the condition you want to start monitoring as a parameter of the **startcondresp** command.

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 91. 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 node(s):
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, refer to the **startcondresp** command man page and "How do I determine the target nodes for a command?" on page 78.

To prevent a user from stopping monitoring, you can lock the condition/response link (as described in "Locking and unlocking conditions, responses, and condition/response links" on page 140). Locking a condition/response link also prevents accidental removal of the link.

For detailed syntax information on the **startcondresp** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and in *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Stopping condition monitoring

The **stopcondresp** command stops monitoring of a condition that has one or more linked responses.

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 89.

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 89.)

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 node(s):
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, refer to the **stopcondresp** command man page and "How do I determine the target nodes for a command?" on page 78.

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, refer to "Locking and unlocking conditions, responses, and condition/response links" on page 140.

For detailed syntax information on the **stopcondresp** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and in *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Removing a condition/response association

The **rmcondresp** command removes the association between a 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 91. The **rmcondresp** command enables you to remove a specified condition/response association, all the associations for a specified condition or all the associations for a specified response.

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 node(s):
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, refer to the **rmcondresp** command man page and "How do I determine the target nodes for a command?" on page 78.

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, refer to "Locking and unlocking conditions, responses, and condition/response links" on page 140.

For detailed syntax information on the **rmcondresp** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Using the audit log to track monitoring activity

When you are monitoring a condition, you should 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, all of the predefined responses that we provide use response scripts that do one of the following:

- logs information to a file

- mails the information to a particular user ID
- broadcasts 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" and the **lsaudrec** command is described in "Listing records from the audit log" on page 100.

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.

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 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 Node(s):

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, refer to the **lsevent** command man page and "How do I determine the target nodes for a command?" on page 78.

For detailed syntax information on the **lsevent** command, refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Listing records from the audit log

To list records from the audit log, use the **lsaudrec** command.

Example: To list all records in the audit log, enter:

```
lsaudrec
```

Output will be similar to the following:

```
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 142. 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 Node(s):

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, refer to the **lsaudrec** command man page and "How do I determine the target nodes for a command?" on page 78.

For detailed syntax information on the **lsaudrec** command, refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 Node(s):

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, refer to the **rmaudrec** command man page and "How do I determine the target nodes for a command?" on page 78.

For detailed syntax information on the **lsaudrec** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 days RMC should keep records in the audit log. Records older than the number of days indicated are automatically deleted by RMC. If the **RetentionPeriod** attribute value is set to 0, this indicates that audit log records should not ever 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 *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*). To list the attribute settings for the IBM.AuditLog instance that represents the the ERRM audit log, use the selection string `-s 'Name == "ERRM"'`. For example:

```
lsrsrc -s 'Name == "ERRM"' IBM.AuditLog
```

This selection string is necessary since other subsystems may have their own audit logs. The preceding command will return output similar to the following.

```
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 deleted 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. For example, 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=3
```

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. Since RSCT subsystems make extensive use of the **/var** file system, 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` provided by the File System Resource Manager. If you are a Cluster Systems Management (CSM) customer, you can also use the predefined condition `AnyNodeVarSpaceUsed` provided by the Domain Management Server Resource Manager. The Domain Management Server Resource Manager is only provided as part of CSM. The `AnyNodeVarSpaceUsed` condition monitors the **/var** file system on all nodes of the management domain.

For more information on the **lsrsrc** and **chrsrc** commands, refer their online man pages. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Advanced resource monitoring

As described in “Basic resource monitoring” on page 88, 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 93, 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, refer to “Creating, modifying and removing conditions.”
- 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, refer to “Creating, modifying, and removing responses” on page 117.

Once you know how to create conditions and responses, be aware that, to be effective, you need to link the conditions together with one or more responses and start monitoring. These tasks are described in “Creating a condition/response association” on page 93 and “Starting condition monitoring” on page 94. For detailed syntax information on any of the commands, refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 89, 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 30 explains each line of the **lscondition** command output shown above.

*Table 30. 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 94 and "Stopping condition monitoring" on page 95.
`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.
`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 93), and start monitoring (as described in "Starting condition monitoring" on page 94), 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.

Table 30. Explanation of ***lscondition*** *command output (continued)*

This line of the lscondition command output...	Indicates...	Notes
`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 142.	Specified by the **-s** flag of both the **mkcondition** and **chcondition** commands.
`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, refer to "The monitoring scope of a condition" on page 75.	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, refer to "The monitoring scope of a condition" on page 75.	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, you should 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, refer to “Listing conditions” on page 89. If you have additional resource managers provided by other products, such as the Cluster Systems Management (CSM) product which provides the Domain Management Server resource manager, refer to that product’s documentation for information on any additional predefined conditions. If you are lucky, there is already a predefined condition that will monitor either the exact event you are interested in, or an event very similar.

Table 31 describes how to proceed when an existing condition exactly suits your needs or is very similar to what you want, or when there are no existing conditions that are similar to what you need.

Table 31. 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, refer to “Creating a condition/response association” on page 93 and “Starting condition monitoring” on page 94.
There is a predefined 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. Refer to “Creating a condition by copying an existing one” on page 109 for more information.
There is no predefined condition that is similar to the event you want to monitor	You will need to define the condition completely from scratch. You will need to examine the available resource managers to see if 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 to be run by RMC to retrieve the value you want to monitor. Refer to “Creating a condition from scratch” on page 111.

Targeting Node(s):

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 using the **mkcondition** command on a CSM management server, do not specify the **-p** flag if you want the condition to be defined on the management server.

For more information, refer to the **mkcondition** command man page and "How do I determine the target nodes for a command?" on page 78.

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, say 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 89, shows that there is a predefined condition named `"/var space used"`. To get detailed information about this predefined condition, you 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 30 in "Creating, modifying and removing conditions" on page 105) 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 116), 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, refer to "Locking and unlocking conditions, responses, and condition/response links" on page 140.

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"
```

Targeting Node(s):
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, refer to the **mkcondition** command man page and "How do I determine the target nodes for a command?" on page 78.

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, let's say 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 142. 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: Usually, the predefined conditions we provide will meet your monitoring needs with, at most, minor modifications. However, if no existing condition is similar to the only you want to create, you need to define the condition completely. To do this, you will need to understand the basic attributes of a condition. Refer to Table 30 in "Creating, modifying and removing conditions" on page 105 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, you can use the following steps to create a condition. There is a significant amount of information you'll need to provide to the **mkcondition** command when defining a condition from scratch. The steps that follow are ordered so that you can carefully consider the purpose and implications of each piece of information you need to supply.

Do the following 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, refer to "What resource managers are provided with RSCT?" on page 67.

If you have additional resource managers provided by other products, such as the Cluster Systems Management (CSM) product which provides the Domain Management Server resource manager, refer to that product's documentation for information on additional resource classes and what attributes they enable you to monitor. You can also examine the available resource classes and attributes using RMC commands (such as the **lsrsrc** command). Refer to "How do RMC and the resource managers enable you to manage resources?" on page 72 for more information on RMC commands. For complete syntax information on the commands, refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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. Refer to "Creating event sensor commands for monitoring" on page 127 for more information.

For example, let's say 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 87), 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 109. 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, you need to 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 142.

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.

   In our example, we’ll just monitor the local node on which the condition is defined. Since this is the default behavior, we will not need to use the **-m** flag.

   For more information on domains in a cluster, refer to “What are management domains and peer domains?” on page 1. For more information on the **-m** flag, refer to the **mkcondition** command’s online man page. Detailed syntax information is also available in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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, refer to"Locking and unlocking conditions, responses, and condition/response links" on page 140.

For detailed syntax information on the **mkcondition** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying a condition

To modify a condition, you use the **chcondition** command. The **chcondition** command uses the same flags as the **mkcondition** command, so it is simply a matter of supplying 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, refer to "Locking and unlocking conditions, responses, and condition/response links" on page 140.

For detailed syntax information on the **chcondition** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 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
```

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, refer to “Locking and unlocking conditions, responses, and condition/response links” on page 140.

Targeting Node(s):
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, refer to the **rmcondition** command man page and “How do I determine the target nodes for a command?” on page 78.

For detailed syntax information on the **rmcondition** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Creating, modifying, and removing responses

There are three commands 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 90, 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 32 describes each line of **lsresponse** command output in more detail.

Table 32. 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 **mkreponse** 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.

Table 32. Explanation of **lsresponse** *command output (continued)*

This line of the lsresponse command output...	Indicates...	Notes
`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.
`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, refer to "Using the audit log to track monitoring activity" on page 97.	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, refer to "Using the audit log to track monitoring activity" on page 97.	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, refer to 34 in "Creating new response scripts" on page 122.)	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 22 in "Predefined and user-defined responses" on page 76.

Creating a response

To create a response, you use the **mkresponse** command. Before creating one, however, you should first check to see if any of the predefined responses that RSCT provides are suitable for your purposes. Refer to Table 22 in "Predefined and user-defined responses" on page 76. For instructions on listing the predefined responses available on your system, refer to "Listing responses" on page 90. If you find a predefined response that does what you need. then you do not need to preform the advanced task of creating a response; instead, you can refer to "Creating a condition/response association" on page 93 and "Starting condition monitoring" on page 94.

Once you understand the information contained in Table 32 in "Creating, modifying, and removing responses" on page 117, 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 33 describes the predefined action scripts, which are located in the directory **/usr/sbin/rsct/bin/**.

   Table 33. 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 33 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, refer to "Creating new response scripts" on page 122.

If you decide to use one of our predefined action scripts, be sure you understand exactly what the script will do. For more information on a script, refer to the script's online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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'll need to provide the full path name of the script and any parameters you need or want to pass it.

Example: Let's say 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 AM and 5 PM 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, refer to “Locking and unlocking conditions, responses, and condition/response links” on page 140.

To add additional actions to a response, use the **chresponse** command, as described in “Modifying a response” on page 125.

Targeting Node(s):

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 CSM management server, do not specify the **-p** flag if you want the condition to be defined on the management server.

For more information, refer to the **mkresponse** command man page and “How do I determine the target nodes for a command?” on page 78.

For detailed syntax information on the **mkresponse** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 33 in “Creating a response” on page 119). 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, say 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 $PERL -e \
'
use POSIX qw(strftime);
print strftime("%A %D %T", localtime($ENV{seconds}) );

'
)

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 34 describes the ERRM environment variables that you can use in response scripts.

Table 34. Event Response Resource Manager environment variables

This environment variable...	Contains...
ERRM_ATTR_NAME	The display name of the attribute used in the expression that caused this event to occur.
ERRM_ATTR_PNAME	The programmatic name of the attribute used in the expression that caused this event to occur.
ERRM_COND_HANDLE	The resource handle (six hexadecimal integers that are separated by spaces and written as a string) of the condition that caused the event.
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, critical. All other severity attribute values are represented in this environment variable as a decimal string.
ERRM_COND_SEVERITYID	The severity value of the condition that caused the event. This environment variable will have one of the following values: 0 (Informational), 1 (Warning), or 2 (Critical).
ERRM_DATA_TYPE	The RMC ct_data_type_t of the attribute that changed to cause this event. The following is a list of valid values for this environment variable: CT_INT32, CT_UINT32, CT_INT64,CT_UINT64, CT_FLOAT32, CT_FLOAT64, CT_CHAR_PTR, CT_BINARY_PTR, and CT_SD_PTR. The actual value of the attribute is stored in the **ERRM_VALUE** environment variable (except for attributes with a data type of CT_NONE).
ERRM_ER_HANDLE	The Event Response resource handle (six hexadecimal integers that are separated by spaces and written as a string) for this event.
ERRM_ER_NAME	The name of the event that triggered this event response script.
ERRM_EXPR	The condition event expression or rearm event expression that tested true, thus triggered this linked response. The type of event that triggered the linked response is stored in the **ERRM_TYPE** environment variable.
ERRM_NODE_NAME	The host name on which this event or rearm event occurred.
ERRM_NODE_NAMELIST	A list of host names. These are the hosts on which the monitored resource resided when the event occurred.
ERRM_RSRC_CLASS_PNAME	The programmatic name of the resource class containing the attribute that changed, thus causing the event to occur.
ERRM_RSRC_CLASS_NAME	The display name of the resource class containing the attribute that changed, thus causing the event to occur.
ERRM_RSRC_HANDLE	The resource handle of the resource whose state change caused the generation of this event (written as a string of six hexadecimal integers that are separated by spaces).
ERRM_RSRC_NAME	The name of the resource whose attribute changed, thus causing this event.
ERRM_RSRC_TYPE	The type of resource that caused the event to occur. This environment variable will have one of the following values: 0 (an existing resource), 1 (a new resource), or 2 (a deleted resource).
ERRM_SD_DATA_TYPE	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.

Table 34. Event Response Resource Manager environment variables (continued)

This environment variable...	Contains...
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.
ERRM_TYPE	The type of event that occurred. The two possible values for this environment variable are *event* or *rearm event*.
ERRM_TYPEID	The value of **ERRM_TYPE**. This environment variable will have one of the following values: 0 (Event) or 1 (Rearm Event).
ERRM_VALUE	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.

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, you should test them before using them 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), refer to the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 119.

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'll 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 Node(s):
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, refer to the **chresponse** command man page and "How do I determine the target nodes for a command?" on page 78.

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, refer to "Locking and unlocking conditions, responses, and condition/response links" on page 140.

For detailed syntax information on the **chresponse** command, refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Removing a response

The **rmresponse** command enables you to remove a response.

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
```

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, refer to "Locking and unlocking conditions, responses, and condition/response links" on page 140.

Targeting Node(s):
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, refer to the **chresponse** command man page and "How do I determine the target nodes for a command?" on page 78.

For detailed syntax information on the **rmresponse** command, refer to its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 system by creating a *sensor*. A *sensor* is merely 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 and/or you can run it explicitly. 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. You can then create a condition to monitor these dynamic attributes that you have defined.

Do the following to create a sensor and a condition to monitor a dynamic attribute returned by the sensor:

1. **Identify a variable value that none of the existing resource managers currently return.** For example, say 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'll 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:

   - Since 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 Sensor resource class. The sensor command script associated with this resource is executed at set intervals and/or when you issue the **refsensor** command. Any value returned by the script is stored as a dynamic attribute of the sensor resource. In our example, the sensor resource is named **NumLogins**, and (since its associated command script contains the statement `print 'Int32='scalar(@output)`,

"\n";) the value we're interested will be available in the **Int32** dynamic attribute. So the following condition will trigger an event if any users are logged into the system.

```
mkcondition -r IBM.Sensor -e "Int32 != 0" -d "users logged in" "users online"
```

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, and so you can also create a condition based on this.

For detailed syntax information on the **mksensor** and **refsensor** commands, refer to their online man pages. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*. These references also contain information on the related sensor commands: **lssensor** (list sensors), **chsensor** (modify a sensor), and **rmsensor** (remove a sensor).

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, refer to 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 35.

Table 35. 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).

Table 35. Terminology differences between the CIM and RMC data models (continued)

This CIM term...	Is analogous to this RMC term...	And refers to...
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.

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 139.

For a current list of CIM classes supported by the CIM resource manager, refer to 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 36 indicates how to locate the MOF files on Linux nodes and AIX nodes.

Table 36. 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. `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 5L 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, refer to the *AIX 5L: 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 AIX 5L, the MOF files will be located in the directory **/usr/pegasus/provider/mof**.
   - On SLES 9, the MOF files will be located in the directory **/usr/share/cmpi/mof** and will have an extension of `.mof`.
   - On 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 5L, 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 **lssrc** command. Use the **lssrc** command's **-s** flag to indicate that you want the status of the CIM resource manager (IBM.CIMRM).
   ```
   lssrc -s IBM.CIMRM
   ```
   Output will be similar to the following. 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 **lssrc** command. Use the **lssrc** command's **-s** flag to indicate that you want the status of the RMC subsystem (ctrmc).
   ```
   lssrc -s ctrmc
   ```
   Output will be similar to the following. 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, issue the **lsrsrc** command without any options to list all resource classes.

```
lsrsrc
```

Output will be similar to the following. (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 **mkcimreg**, **lsrsrc**, and **rmcctrl** commands, refer to their online man pages. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 130 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
```

Output will be similar to the following:

```
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
```

Output is similar to the following:

```
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, refer their online man pages. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 108) 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 93) 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 94.

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, refer their online man pages. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 130 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
```

Output will be similar to the following:

```
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
```

Output will be similar to the following:

```
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
```

Output will be similar to the following:

```
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 37 on page 138.

Table 37. 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 37 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 Node(s):

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, as described in “Registering CIM classes and providers” on page 130, 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.

Note that for Linux, methods are supported by a subset of Linux SLES9 providers only. There is no method support among AIX OS base providers.

Catching SNMP traps on Linux nodes

Note: 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 119.

   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, refer to their online man pages. Detailed syntax information is also available in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Locking and unlocking conditions, responses, and condition/response links

Conditions, responses, and condition/response links can be locked to prevent user modification or removal. A locked condition, response, or condition/response link cannot be modified or removed until it is unlocked. For this reason, the following commands for manipulating 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, response, or condition/response link is locked. The commands that will fail to act upon a particular locked resource are:

- 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 **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, response, or condition/response link, while the **-U** flag unlocks it.

Before using the **-U** flag, you should be aware that if a particular 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, you should exercise caution before unlocking a condition, response, or condition/response link. In general, if you do not know why the monitoring resource is locked, you should 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*]

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*]

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*

Examples:

- Say 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 73 and “Rearm event expressions” on page 74.

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 38 summarizes the commands that accept a selection string expression. For general information about how the selection strings are used by these commands, refer to the topics referenced in the table.

Table 38. Commands whose actions you can restrict using selection strings

Command	Purpose	Selection string expression action	For more information, refer to...
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 116
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.	*Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*
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 97
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.	*Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*
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 108
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 102
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.	*Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*

You can also find complete syntax information on any of these commands in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

SQL restrictions

SQL syntax is supported for selection strings. Table 39 relates the RMC terminology to SQL terminology.

Table 39. 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 40 lists the base data types supported by the RMC subsystem for variables and constants in an expression.

Table 40. 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 simply *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" on page 144 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

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 0x01020304050607090a0b0c0d0e0f1011121314"
```

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 145, 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 zero, 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, LIKE, and so on). 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 41 summarizes the set of operators that can be used in expressions.

Table 41. Operators that can be used in expressions

Operator	Description	Left data types	Right data types	Example	Notes
+	Addition	Integer,float	Integer,float	"1+2" results in 3	None
-	Subtraction	Integer,float	Integer,float	"1.0-2.0" results in -1.0	None
*	Multiplication	Integer,float	Integer,float	"2*3" results in 6	None
/	Division	Integer,float	Integer,float	"2/3" results in 1	None
-	Unary minus	None	Integer,float	"-abc"	None
+	Unary plus	None	Integer,float	"+abc"	None

Table 41. Operators that can be used in expressions (continued)

Operator	Description	Left data types	Right data types	Example	Notes
..	Range	Integers	Integers	″1..3″ results in 1,2,3	Shorthand for all integers between and including the two values
%	Modulo	Integers	Integers	″10%2″ results in 0	None
|	Bitwise OR	Integers	Integers	″2|4″ results in 6	None
&	Bitwise AND	Integers	Integers	″3&2″ results in 2	None
˜	Bitwise complement	None	Integers	˜0x0000ffff results in 0xffff0000	None
^	Exclusive OR	Integers	Integers	0x0000aaaa^0x0000ffff results in 0x00005555	None
>>	Right shift	Integers	Integers	0x0fff>>4 results in 0x00ff	None
<<	Left shift	Integers	Integers	″0x0ffff<<4″ results in 0xffff0	None
== =	Equality	All but SDs	All but SDs	″2==2″ results in 1 ″2=2″ results in 1	Result is true (1) or false (0)
!= <>	Inequality	All but SDs	All but SDs	″2!=2″ results in 0 ″2<>2″ results in 0	Result is true (1) or false (0)
>	Greater than	Integer,float	Integer,float	″2>3″ results in 0	Result is true (1) or false (0)
>=	Greater than or equal	Integer,float	Integer,float	″4>=3″ results in 1	Result is true (1) or false (0)
<	Less than	Integer,float	Integer,float	″4<3″ results in 0	Result is true (1) or false (0)
<=	Less than or equal	Integer,float	Integer,float	″2<=3″ results in 1	Result is true (1) or false (0)
=~	Pattern match	Strings	Strings	″abc″=~″a.*″ results in 1	Right operand is interpreted as an extended regular expression. To use this operator in an expression, the locale(s) of the node(s) running the RMC daemon must be using either Unicode Transfer Format-8 (UTF-8) encoding (or a codeset that matches UTF-8), or else C locale encoding. If multiple nodes are involved, the encoding must be consistent across all nodes.
!~	Not pattern match	Strings	Strings	″abc″!~″a.*″ results in 0	Right operand is interpreted as an extended regular expression. To use this operator in an expression, the locale(s) of the node(s) running the RMC daemon must be using either Unicode Transfer Format-8 (UTF-8) encoding (or a codeset that matches UTF-8), or else C locale encoding. If multiple nodes are involved, the encoding must be consistent across all nodes.
=? LIKE like	SQL pattern match	Strings	Strings	″abc″=? ″a%″ results in 1	Right operand is interpreted as a SQL pattern

Table 41. Operators that can be used in expressions (continued)

Operator	Description	Left data types	Right data types	Example	Notes
!? NOT LIKE not like	Not SQL pattern match	Strings	Strings	″abc″!? ″a%″ results in 0	Right operand is interpreted as a SQL pattern
|< IN in	Contains any	All but SDs	All but SDs	″{1..5}|<{2,10}″ results in 1	Result is true (1) if left operand contains any value from right operand
>< NOT IN not in	Contains none	All but SDs	All but SDs	″{1..5}><{2,10}″ results in 0	Result is true (1) if left operand contains no value from right operand
&<	Contains all	All but SDs	All but SDs	″{1..5}&<{2,10}″ results in 0	Result is true (1) if left operand contains all values from right operand
|| OR or	Logical OR	Integers	Integers	″(1<2)||(2>4)″ results in 1	Result is true (1) or false (0)
&& AND and	Logical AND	Integers	Integers	″(1<2)&&(2>4)″ results in 0	Result is true (1) or false (0)
! NOT not	Logical NOT	None	Integers	″!(2==4)″ results in 1	Result is true (1) or false (0)

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 42 shows the default precedence order for operators in expressions. All operators in the same table cell have the same or equal precedence.

Table 42. Operator precedence in expressions, from highest to lowest

Operators	Description
.	Structured data element separator
~	Bitwise complement
! NOT not	Logical not
–	Unary minus

Table 42. Operator precedence in expressions, from highest to lowest (continued)

Operators	Description
+	Unary plus
*	Multiplication
/	Division
%	Modulo
+	Addition
–	Subtraction
<<	Left shift
>>	Right shift
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
==	Equality
!=	Inequality
=? LIKE like	SQL match
!?	SQL not match
=~	Regular expression match
!~	Regular expression not match
?=	Regular expression match (compat)
|< 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.

Examples of expressions

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%0")&&(IntList|<(1,3,5..7))
   (Name=~'tr.*0') AND (IntList IN {1,3,5..7})
   ```

Chapter 5. 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.

Example: For example, a non-root user on a CSM 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, say *user01*, 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, on AIX or Linux. The nodes may be independent workstations, or may be in a management or 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. Administrators may use not only resource monitoring and control (RMC) commands to manage LP resources, but also the following LPRM commands.

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 164.
- 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, refer to "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 (as described in "Managing user access to resources using RMC ACL files" on page 80), 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 will be ignored.

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 43 provides more details concerning the four types of LP ACLs.

Table 43. 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 43. 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 the forms described in Table 44 on page 157.

Table 44. LP ACL user identifier forms and specifications

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 (described in Chapter 7, “Understanding and administering cluster security services,” on page 205). 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 forms.	
	This host user identifier format...	**Specifies...**
	user_name@*host_identifier*	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	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.
	*	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 229 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 refer to “Specifying Host Based Authentication ACL entries” on page 173.

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 an RSCT peer domain or a CSM management domain, use the 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(s) specified by the user identifier. These permissions are described in Table 45.

Table 45. User permissions in an LP ACL entry

Specifying this...	**Indicates that the specified user(s) at the specified host(s) have...**
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.

Table 45. User permissions in an LP ACL entry (continued)

Specifying this...	**Indicates that the specified user(s) at the specified host(s) 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:	
	Specifying this...	**Indicates that the specified user(s) at the specified host(s) have...**
	q	Query permission. This allows the user to query persistent or dynamic attributes.
	l	List permission. This allows the user to list resources.
	e	Event permission. This allows the user to register, query, and unregister events.
	v	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:	
	Specifying this...	**Indicates that the specified user(s) at the specified host(s) have...**
	d	Define permission. This allows the user to define and undefine resources.
	c	Refresh permission. This allows the user to refresh resource configuration.
	s	Set permission. This allows the user to set attributes.
	o	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, refer to the **lpacl** man page or the **lpacl** entry in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

| ## Permissions for LPRM commands

Table 46 outlines the permissions that are required to run the LPRM commands.

Table 46. Permissions required to run the LP resource manager commands

LPRM command		**Class ACL**	**Resource ACL**	**Resource Initial ACL**
mklpcmd	(no ACL entries specified)	w		
	(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, refer to the man pages or to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 211 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 rootcmd
Resource 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 rootcmd
Name = 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 rootcmd
Name = 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]# lslpclacl
   Class 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:

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.

Further details about these commands appear in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 105.

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:
- “What resource managers are provided with RSCT?” on page 67
- “Using the audit log to track monitoring activity” on page 97

Defining LP resources and authorized users

Use the **mklpcmd** command to create an LP resource, and, optionally, to set permissions for an LP resource.

Before you begin:
- You need the following permissions 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.

Action	Required Class ACL permission	Required Resource Initial ACL permission
Create LP command without specifying permissions	**w**	
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 168) and the Resource Initial ACL (as described in “Displaying the Resource Initial ACL” on page 171).
- If you are going to specify the optional set of permissions for the resource you are creating, you should understand the concepts described in “Overview of the LP resource manager’s access control lists” on page 154.

- You need to 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 164.

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. You should decide whether 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 170.
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.

For complete syntax information on the **mklpcmd** command, refer to the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*

Running an LP resource

Use the **runlpcmd** command to run a root command or script that is defined as an LP resource.

Before you begin:

- You need the following permissions 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.

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 168) and the Resource ACL (as described in “Displaying a Resource ACL” on page 169).

- You need to 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 164.

Perform the following step to run an LP resource.

1. 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, refer to the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*

Modifying an LP resource

Use the **chlpcmd** command to modify an LP resource.

Before you begin:

- You need the following permissions 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.

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 168) and the Resource ACL (as described in “Displaying a Resource ACL” on page 169).

- You need to 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 164.

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.

For complete syntax information on the **chlpcmd** commands, refer to the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*

Removing LP resources

Use the **rmlpcmd** command to remove an LP resource.

Before you begin:

- 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, check the setting in the Class ACL as described in "Displaying the Class ACL").
- You need to 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 164.

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, refer to the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*

Displaying the Class ACL

Use the **lslpclacl** command to display the access controls for the IBM.LPCommands resource class.

Before you begin:

- 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 154.
- You need to 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 164.

To display the Class ACL, issue the **lslpclacl** command.

Example:

```
lslpclacl
```

Result: Output will be similar to the following:

```
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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying the Class ACL

Use the **chlpclacl** command to change the access controls for the IBM.LPCommands resource class.

Before you begin:

- To modify the Class 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. To determine if you have the correct permissions, check the settings in the Class ACL as described in “Displaying the Class ACL” on page 168).
- Understand the purpose of the Class ACL as described in “Overview of the LP resource manager’s access control lists” on page 154.
- You need to 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 164.

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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Displaying a Resource ACL

Use the **lslpracl** command to display the access controls for a resource of the IBM.LPCommands resource class.

Before you begin:

- To display a Resource ACL, you need the following permissions specified in the Class ACL and Resource ACL. Alternatively, you can display 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	**r**

- Understand the purpose of a Resource ACL as described in “Overview of the LP resource manager’s access control lists” on page 154.
- You need to 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 164.

To display the Resource ACL, issue the **lslpracl** command with the name of the resource.

Example:

```
lslpracl LPCommand1
```

Result: Output will be similar to the following:

```
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, refer to “Displaying a Resource Shared ACL” on page 172 or issue the **lslpracl** command with its **-L** flag.

Example:

```
lslpracl -L LPCommand1
```

For complete syntax information on the **lslpracl** command (including additional flags you can specify), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying a Resource ACL

Use the **chlpracl** command to change the access controls for a resource of the IBM.LPCommands resource class.

Before you begin:

- 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 154.
- You need to 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 164.

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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Displaying the Resource Initial ACL

Use the **lslpriacl** command to display the access controls for the Resource Initial ACL.

Before you begin:

- To display the Resource Initial 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 Resource Initial ACL as described in "Overview of the LP resource manager's access control lists" on page 154.
- You need to 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 164.

To display the Resource Initial ACL, issue the **lslpriacl** command.

Example:

```
lslpriacl
```

Result: Output will be similar to the following:

```
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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying the Resource Initial ACL

Use the **chlpriacl** command to change the access controls for the Resource Initial ACL.

Before you begin:

- To modify the Resource Initial 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 Resource Initial ACL as described in “Overview of the LP resource manager’s access control lists” on page 154.
- You need to 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 164.

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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Displaying a Resource Shared ACL

Use the **lslprsacl** command to display the access controls for Resource Shared ACL.

Before you begin:

- To display the Resource Shared 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 Resource Shared ACL as described in “Overview of the LP resource manager’s access control lists” on page 154.
- You need to 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 164.

To display the Resource Shared ACL, issue the **lslprsacl** command.

Example:

```
lslprsacl
```

Result: Output will be similar to the following:

```
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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Modifying a Resource Shared ACL

Use the **chlprsacl** command to change the access controls for the Resource Shared ACL.

Before you begin:

- 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 154.
- You need to 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 164.

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), refer to its online man page. Detailed syntax information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Specifying Host Based Authentication ACL entries

As described in “Overview of the LP resource manager’s access control lists” on page 154, LP ACL entries may specify network identifiers provided by the Host Based Authentication (HBA) security mechanism. 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 CSM 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 simply 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 rootcmd
Resource 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 rootcmd
Resource 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 re
source '0x6040 0xffff 0xbb5dca56 0xfe5aa5e8 0x0f6b6f40 0xbf598b90' of class I
BM.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 rootcmd
Resource 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 rootcmd
Resource 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 rootcmd
Resource 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 179 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_bld
Resource 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_prod
Resource 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_any
Resource 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 3 in "Restricted execution based on command arguments" on page 177. 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
```

Simply instructing everybody to execute the following 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 177, 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 3 in "Restricted execution based on command arguments" on page 177. 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 whose Name attribute is *rpower_bld*, the filter script for that resource would not allow it, since *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, since *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.

Chapter 6. Understanding and 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 manger 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 manger 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 does the Storage resource manager gather and represent 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, will automatically *harvest* (detect and collect) information about locally attached physical disks and their related, logical, storage entities. The information collected will be mapped to instances of the Storage resource manager 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 on Linux and physical volumes which are sole members of a volume group on AIX.

Volume group resource (IBM.VolumeGroup)
This resource class externalizes the attributes of volume groups comprised of only one physical volume on AIX nodes. Volume group entities are not harvested on Linux nodes, but AIX volume group resources may still be managed from Linux nodes in a heterogeneous cluster.

Logical volume resource (IBM.LogicalVolume)
This resource class externalizes the attributes of logical volumes configured on AIX volume groups. The logical volume entities are not harvested on Linux nodes, but AIX logical volume resources may still be managed from Linux nodes in a heterogeneous cluster.

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) 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 of differences between how the storage entities are related on AIX and Linux nodes, the way these entities are mapped to the resource classes is similarly different.

On Linux nodes, disk resources contain partition resources which contain file system resources. These relationships are mapped to IBM.Disk, IBM.Partition, and IBM.AgFileSystem resources, as illustrated in Figure 4.

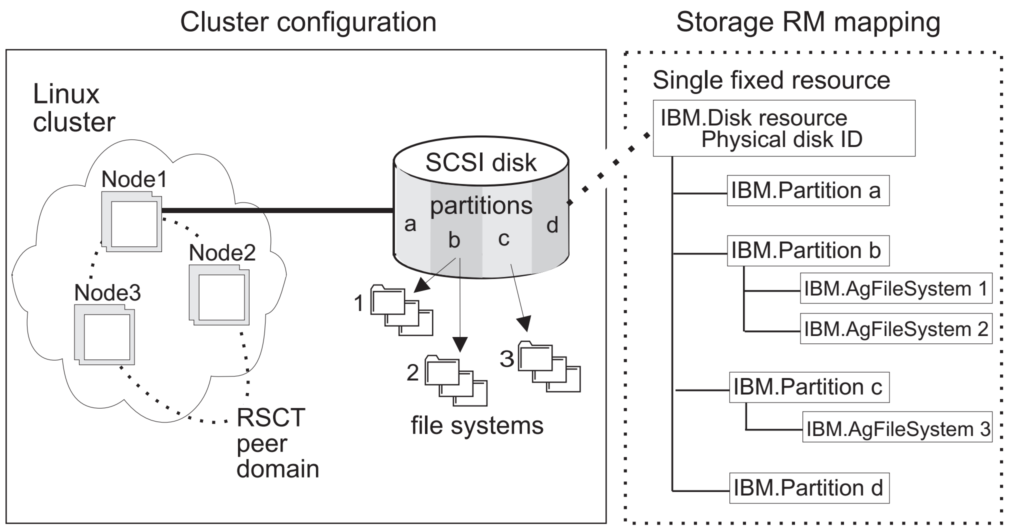

Figure 4. Mapping of storage to Storage RM resources on Linux

The relationship between the storage entities 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” on page 201 for more information.

On AIX nodes, volume group resources (each dependent on one disk resource) 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 on page 185.

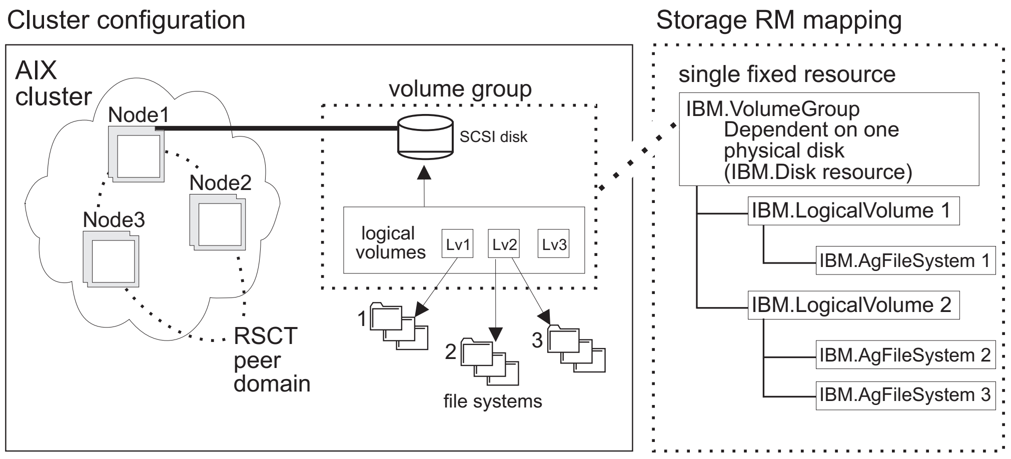

Figure 5. Mapping of storage to Storage RM resources on AIX

The relationship between the storage entities 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 nodes” on page 203 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 if 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. In this case, the fixed resources that all represent the same storage entity are called *constituent resources* of the *aggregate resource*.

Figure 6 on page 186 illustrates the storage mapping for an aggregate disk resource and its constituents, shared across two nodes in an RSCT peer domain.

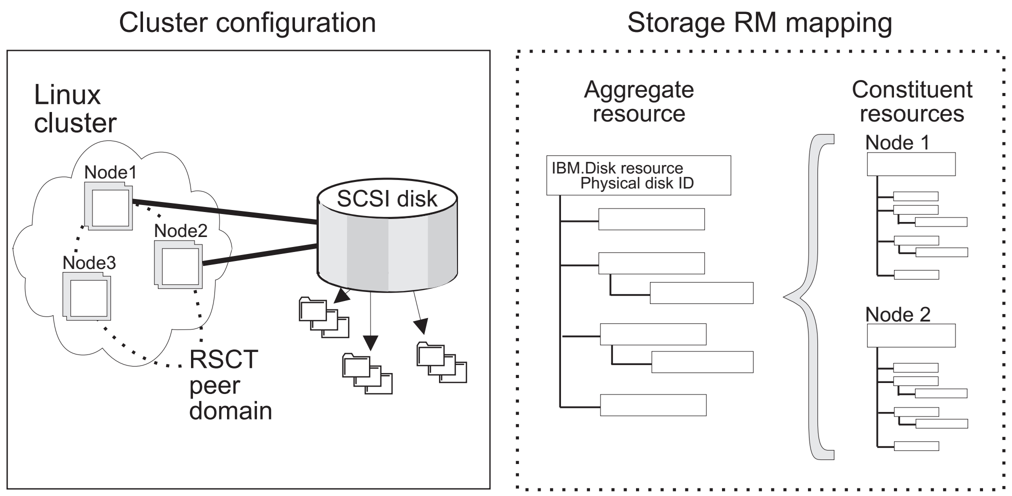

Figure 6. Storage mapping for an aggregate disk resource and its constituents

Having an aggregate resource simplifies administration, as you can issue commands against the aggregate resource to affect its constituents. This enables 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 197, 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 aggregate resource's constituent resources.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, 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 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 if a resource is a single-fixed, constituent, or aggregate resource" on page 200 for more information.

In addition to harvesting storage information when a peer domain is brought online, the Storage resource manager will, at regular intervals, repeat the harvest process in order to detect changes in the peer domain's storage configuration (such as 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 194.

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 may no longer represent an actual storage entity. Instead of being deleted, it is, by default, simply 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 simply marked as ghost resources. See “Enabling automatic deletion of harvested resources when they are not detected by subsequent harvest operations” on page 195 for more information.

“Listing resource information” on page 83 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, refer to their online man pages. Detailed information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

How does the Storage resource manager monitor 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). 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) to 0 (disabled). See “Disabling disk locking” on page 195.

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_online_*** subroutines. For each particular resource class, the terms *online* and *offline* have unique meanings depending on the 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 will, at intervals determined by the resource class' MonitorInterval property, check the status of resource to make sure it is still online. For the IBM.Disk resource class, for example, this means passively re-reserving the disk. For the IBM.AgFileSystem resource class, this means checking the status of the IBM.AgFileSystem resource to make sure it's still mounted. 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, refer to the *Reliable Scalable Cluster Technology: RMC Programming Guide and Reference*. For information on resource monitoring using the command line, refer to Chapter 4, “Managing and monitoring resources using RMC and resource managers,” on page 63.

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 disk locking has not been disabled, also result in the IBM.Disk resource being brought online (in other words, reserved using SCSI reserves). 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, mounting a file system represented by a harvested IBM.FileSystem resource using the **startrsrc** command or the **mc_online_*** subroutines, will not be successful unless the associated disk is, or can be, reserved by the node (and on AIX nodes, only if the associated volume group is, or can be, activated).

Table 47 describes the online and offline operations for the resource classes available on Linux nodes, 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 status of 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 195).

Table 47. Disk reservation and release summary (Linux)

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 it is not already reserved by another node).	passively re-reserve the disk.	release the disk (provided none of the IBM.Partition or IBM.AgFileSystem resources that it contains are online).

Table 47. Disk reservation and release summary (Linux) (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.Partition	provided the associated disk is not reserved by another node, will perform the online operation on the associated disk. 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.	provided no IBM.AgFileSystem resource associated with the IBM.Partition resource is online, the IBM.Partition resource will be considered offline. Will also perform the offline operation on the IBM.Disk resource (provided no other IBM.Partition resource associated with the IBM.Disk resource is online).
IBM.AgFileSystem	will 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, will 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 no other IBM.AgFileSystem resource associated with the IBM.Partition resource is online).

Table 47 on page 188 assumes that the DeviceLockMode attribute of the IBM.Disk resource is set to the default (DeviceLockMode=1). If device locking is disabled for the disk (DeviceLockMode=0), then it will not be reserved or monitored. 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 48 describes the online and offline operations for the resource classes available on AIX nodes, 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 status of 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 195).

Table 48. Disk reservation and release summary (AIX)

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 it is not already reserved by another node).	passively re-reserve the disk.	release the disk (provided no IBM.VolumeGroup thator IBM.AgFileSystem resources that it contains are online).

Table 48. Disk reservation and release summary (AIX) (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	will perform the online operation on the IBM.Disk resource on which the volume group depends. If the online operation is successful for all IBM.Disk resource, will activate the volume group (also using SCSI reserves).	check that the volume group is active.	provided no IBM.LogicalVolume resource is associated with the IBM.VolumeGroup is online, will deactivate the volume group. Will also perform offline operation on the IBM.Disk resource on which the volume group depends.
IBM.LogicalVolume	Will 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.	provided no IBM.AgFileSystem resource associated with the IBM.LogicalVolume resource is online, the IBM.LogicalVolume resource will be considered offline. Will also perform the offline operation on the IBM.VolumeGroup resource (provided no other IBM.LogicalVolume resource associated with the IBM.VolumeGroup resource is online).
IBM.AgFileSystem	will 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, will 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 no other IBM.AgFileSystem resource associated with the IBM.LogicalVolume resource is online).

Table 48 on page 189 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.

Storage resource manager requirements

Before using the Storage resource manager, make sure your system meets the prerequisites described in "Storage resource manager prerequisites" on page 191, and you perform the necessary configuration described in "Storage resource manager configuration requirements" on page 191.

Storage resource manager prerequisites

The following software, hardware, and configuration requirements apply for exploiting Storage resource manager functions:

- IBM TotalStorage DS4000™ series (formerly FAStT family)
- 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. 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 FC RAID (Fibre Channel redundant array of independent disks) device driver should be installed for DS4000 series disk systems. The required AIX filesets are:

- devices.fcp.disk.array.diag
- devices.fcp.disk.array.rte
- devices.fcp.disk.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 AIX logical volume, some configuration is necessary for the Storage resource manager to work effectively. In particular, you must do the following:

- On AIX nodes, configure shared volume groups to contain only a single disk (AIX LVM volume groups containing more than one disk are not recognized by the Storage resource manager), and 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 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 193.

Storage resource manager configuration

Some configuration is needed for file systems and shared volume groups to work with the Storage resource manager on AIX. Before you proceed, be sure to review the information in “Storage resource manager requirements” on page 190.

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 must also reserve the physical volumes. For this reason, the Storage resource manager only harvests volume groups comprised of one physical volume.

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.

For 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 did 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 example:

```
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. 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. For example, to obtain the physical volume ID of the physical volume *hdisk2*, you would enter the command:

   ```
   getlvodm -p hdisk2
   ```

   Output will be the physical volume ID. For example:

   ```
   000000623e576668
   ```

2. 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. For example:
      ```
      lspv
      ```
      Output will be similar to:
      ```
      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 on upon the completion of the volume group import into the system. For 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
```

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 197).

Storage resource manager optional configuration

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 managers 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

As described in “How does the Storage resource manager gather and represent storage data?” on page 183, the Storage resource manager automatically harvests information about physical disks (and related storage entities) and represents this information using its resource classes. 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
```

Command output will be similar to the following:

```
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 **chrsrc** 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

As described in "How does the Storage resource manager gather and represent storage data?" on page 183, 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. A ghost resource represents a Storage resource manager resource that may no longer represent an actual storage entity. It is simply 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

"How does the Storage resource manager monitor the availability of shared storage?" on page 187 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 container resources on recursively so that no online operation is successful unless the Storage resource manager is able to reserve the underlying disk resource.

Although disk locking is the default behavior of the Storage resource manager, you can override this behavior for individual disks by:

- On Linux, setting the IBM.Disk resource's DeviceLockMode to 0. For example, to disable disk locking for the disk *000000150f450a54*, you would enter the following on on node that is online in the peer domain:

  ```
  chrsrc -s "Name=='000000150f450a54'" IBM.Disk DeviceLockMode=0
  ```

- On AIX, setting the IBM.VolumeGroup resource's DeviceLockMode to 0. For example, to disable disk locking for volume group *hd2* and its underlying physical volume, you would enter the following 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

As described in "How does the Storage resource manager monitor the availability of shared storage?" on page 187, the online operation for IBM.AgFileSystem resources is to mount the file system represented by the IBM.AgFileSystem resource.

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 does the Storage resource manager monitor the availability of shared storage?" on page 187, 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. 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, it will by default, attempt to resolve the problem by issuing the **fsck** command, and then will attempt 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 49 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 49. PreOnlineMethod attribute settings for the IBM.AgFileSystem resource class

PreOnlineMethod setting	**File system checking method**	
	Linux	**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 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 50 on page 197 summarizes the PreOnlineMethod attribute settings for an IBM.AgFileSystem resource.

Table 50. PreOnlineMethod attribute settings for an IBM.AgFileSystem resource

PreOnlineMethod setting	File system checking method	
	Linux	**AIX**
0 (default)	Use method determined by the IBM.AgFileSystem resource class.	Use method determined by the IBM.AgFileSystem resource class.
1	No operation	No operation
2	**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 the file system *fs1* by setting its PreOnlineMethod resource attribute to 1.

```
chrsrc -s "Name=='/fs1'" IBM.AgFileSystem PreOnlineMethod=1
```

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 194. 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

As described in “How does the Storage resource manager gather and represent storage data?” on page 183, 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. 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 harvest 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 attributes listed 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 83. Detailed information is also provided in the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* and the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Discerning the relationship between storage resources

The storage model for Linux nodes is different than the storage model for AIX nodes. On Linux nodes, disk resources contain partition resources which contain file system resources. On AIX nodes, volume group resources (which are dependent on disk resources) contain logical volume resources, which contain file system resources.

What's more 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 does the Storage resource manager gather and represent storage data?” on page 183, 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 if a resource is a single-fixed, constituent, or aggregate resource” on page 200 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. The relationships among the storage entities are captured by the resource attributes ContainerResource and ContainerResourceID (for IBM.AgFileSystem, IBM.LogicalVolume, and IBM.Partition resources), and by the resource attributes DependentResource and DependentResourceId (for IBM.Disk resources on AIX nodes). Typically, understanding the relationships explicitly 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. Since the storage model for Linux nodes is different than the storage model for AIX nodes, see “Discerning the relationship between storage entities on Linux nodes” on page 201 or “Discerning the relationship between storage entities on AIX nodes” on page 203 as appropriate.

Determining if a resource is a single-fixed, constituent, or aggregate resource

As described in “How does the Storage resource manager gather and represent storage data?” on page 183, 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)

A resource's ResourceType attribute indicates whether the resource is a fixed resource or an aggregate resource.

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, refer 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.

For example, the following **lsrsrc** command lists the physical disks.

```
lsrsrc IBM.Disk Name ResourceType AggregateResource
```

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 0x
e738f35b
```

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. For 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. You could use similar commands to list constituent resources for aggregates resources of the other Storage resource manager resource classes.

```
lsrsrc -s "AggregateResource=='0x2036 0xffff 0x5f47b7ad 0x2b379874 0x8f9cc90c \
0xe738f35b'" IBM.Disk
```

Discerning the relationship between storage entities on Linux nodes

On Linux nodes, disk resources 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.

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
```

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. For 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
```

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"
```

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 nodes

On AIX nodes, volume group resources are dependent on one disk resource. 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.

The following **lsrsrc** command lists the ContainerResourceId information for all file systems on AIX *node1*:

```
lsrsrc -t -s "NodeNameList=='node1'" IBM.AgFileSystem \
 Name DeviceName ContainerResourceId
```

Command output similar to the following is returned. 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"    "0024a09a00004c0000001091bda12ea.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. For example:

```
lsrsrc -t -s "Name=='/usr' && NodeNameList=='node1'" IBM.AgFileSystem \
 Name DeviceName ContainerResourceId
```

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.

```
lsrsrc -t -s "ResourceId=='0024a09a00004c00000001069d28c253.5' && \
 NodeNameList=='c48f1rp16'" \
 IBM.LogicalVolume Name DeviceName ContainerResourceId
```

Command output similar to the following is returned. 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, we 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 step, we can list the IBM.Disk resource which is a member of the volume group in question.

```
lsrsrc -t -s "DependentResourceId=='0024a09a00004c00000001069d28c253' \
 && NodeNameList=='node1'" \
 IBM.Disk Name DeviceName DeviceInfo
```

Command output similar to the following is returned:

```
Resource Persistent and Dynamic Attributes for IBM.Disk
Name               DeviceName    DeviceInfo
"0024a09a1b6f2012" "/dev/hdisk0" "VG=rootvg VGID=0024a09a00004c00000001069d28c253"
```

Chapter 7. Understanding and administering cluster security services

You can administer cluster security services for both an RSCT peer domain and a management domain. For information on creating an RSCT peer domain, refer to Chapter 3, “Creating and administering an RSCT peer domain,” on page 17. For information on creating a management domain, refer to *IBM Cluster Systems Management: Administration Guide*.

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, various cluster applications (such as the configuration resource manager and CSM) can check that other parties are genuine and not attempting to gain unwarranted access to the system. “Understanding cluster security services authentication” describes how authentication is handled on both an RSCT peer domain and a management domain.

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, described in “Managing user access to resources using RMC ACL files” on page 80, 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*, and is described in “Understanding cluster security services authorization” on page 210.

“Cluster security services administration” on page 214 presents a series of administrative tasks that you may need or want to perform and the steps to perform them.

Understanding cluster security services authentication

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 80.

Cluster security services uses **credential 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.

Credential based authentication involves the use of a third party that both the client and the server trust. For this release, only Host Based Authentication is supported but other security mechanisms may be supported in the future. In the case of Host Based Authentication, the trusted third party is the operating system. This method of authentication is used between RSCT and its client applications (such as CSM).

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 provides an abstraction layer between the cluster components and the underlying security mechanisms. This abstraction layer is called the *mechanism abstraction layer* (MAL) and 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**

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. You should *not* modify this file, but you should be aware that the file exists and is used by cluster security services 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:

- 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, refer to “Changing a node’s private/public key pair” on page 226 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. You should 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 using CSM or as an RSCT peer domain using configuration resource manager commands), the necessary public key exchanges will, by default, be carried out by CSM or configuration resource manager utilities. If the network is relatively secure against identity and address spoofing, you can use these utilities; otherwise, the keys should be transferred manually to prevent to prevent the inclusion of nodes that may be attempting to masquerade as known nodes. You should carefully consider whether the security of the network is sufficient to prevent address and identity spoofing. If you do not think the network is secure enough, refer to “Guarding against address and identify spoofing when transferring public keys” on page 223. 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 51 on page 208 lists the default locations for a node’s private key, public key, and trusted host list.

Table 51. 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.
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. Refer to "Configuring the ctcasd daemon on a node" on page 217 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, refer to "Configuring the cluster security services library" on page 214.

The Enhanced Host Based Authentication mechanism (HBA2)

The Enhanced Host Based Authentication mechanism (HBA2) is available beginning in RSCT 2.3.10.0 and 2.4.6.0. It 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.

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, refer to "Configuring the cluster security services library" on page 214.

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 address(es). 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. This feature is available for RSCT peer domains in which all nodes are running RSCT version 2.3.1.0 or later.

Versions of RSCT prior to 2.3.1.0 do not support IP address authentication. Also, CSM itself does not use IP address authentication and, thus, it does not configure the nodes in a management domain to use it.

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 simply indicate 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.

Understanding cluster security services authorization

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 80, 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” on page 211.

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 212.

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” on page 210, 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, refer to “Configuring the global and local authorization identity mappings” on page 226.

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 212.

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 211, 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

As discussed in "Understanding cluster security services authorization" on page 210, a security context is established in the service application for the client application when the client is authenticated. 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

Alternate authorization mechanisms are introduced in RSCT 2.3.10.0 and 2.4.6.0. This feature permits existing service application owners and administrators to re-use existing access control lists and infrastructures built for use in the former RSCT single MPM environment in an environment where multiple MPMs exist. Prior to 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 re-use 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 instance, 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

As mentioned in "Understanding credentials based authentication" on page 206, the **/usr/sbin/rsct/cfg/ctsec.cfg** configuration file defines the available MPMs. 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.

Cluster security services administration

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

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 52 shows the available MPMs provided by cluster security services and the security mechanism that they support:

Table 52. MPMs provided by cluster security services

MPM Mnemonic	MPM Path name	Supported 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 "Understanding cluster security services authorization" on page 210 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 212. 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 “Understanding cluster security services authorization” on page 210.

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 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, refer its online man page. Detailed syntax information is also provided in *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or in *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Configuring the Host Based Authentication mechanisms

Table 53 on page 217 describes the administrative tasks you may need or want to perform that are related to the Host Based Authentication mechanisms.

Table 53. 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 221	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 223	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 226	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 file sets 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 54 describes the configurable parameters.

Table 54. 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, refer to *Reliable Scalable Cluster Technology for AIX 5L: Diagnosis 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. For more information about tracing the **ctcasd** daemon, refer to *Reliable Scalable Cluster Technology for AIX 5L: Diagnosis Guide*.
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 54. 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 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, refer to *Reliable Scalable Cluster Technology for AIX 5L: Diagnosis 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, refer to *Reliable Scalable Cluster Technology for AIX 5L: Diagnosis 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.

Table 54. Keyword descriptions for the ctcasd daemon configuration file (continued)

Keyword	Description
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, refer to "Configuring credential life span" on page 221.
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, refer to "Configuring credential life span" on page 221.

Table 54. Keyword descriptions for the ctcasd daemon configuration file (continued)

Keyword	Description
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. This entry should not be modified by system administrators 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

As described in “Configuring the ctcasd daemon on a node” on page 217, the **ctcasd.cfg** file’s HBA_CRED_TIMETOLIVE and HBA2_CRED_TIMETOLIVE keywords set the credential life spans for their respective security mechanisms. 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.

You should 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, you should 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: Say you decide 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 55 shows some examples of valid specifications for the HBA_CRED_TIMETOLIVE keyword; the HBA2_CRED_TIMETOLIVE keyword behaves in the same manner.

Table 55. 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 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 the configuration resource manager.

- In a management domain 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, refer to *IBM Cluster Systems Management: Administration Guide*.
- 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, refer to "Preparing the initial security environment on each node that will participate in the peer domain" on page 25.

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 may allow itself to be managed by the wrong "management server," or the wrong "managed node" may be invited into the network.

If you do not feel your network is sufficiently secure to avoid a possible spoofing attack, you should do the following 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. Refer to "Manually transferring public keys" for more information.
- For a management domain, Verify the accuracy of the keys automatically transferred by the CSM **updatenode** and **installnode** commands. See "Verifying the accuracy of keys that have been automatically transferred" on page 225 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 below.
3. Issue the **ctsvhbal** command to obtain the list of available host based authentication mechanism identities for this system. Output from this command would be similar to:

```
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 above 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 have been automatically transferred: When establishing a management domain, CSM's **updatenode** and **installnode** commands will automatically copy:
- 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

As described in “Understanding cluster security services authorization” on page 210, 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). 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 56 describes what you need to do to configure global and local authorization identity mappings.

Table 56. 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 214). 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, refer to "Configuring the Host Based Authentication (HBA) mechanism mappings" on page 229 and "Configuring the Enhanced Host Based Authentication (HBA2) mechanism mappings" on page 232.

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, refer to *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

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 Host Based Authentication mechanism, you must begin the entry with the **unix:** mnemonic.

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 Host Based Authentication 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, you should 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 57 shows several examples of how an entry can use the ***** wildcard character when specifying the user name portion of the security network identifier.

Table 57. 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 Host Based Authentication mechanism network identifier from the host *epsilon3.ibm.com*.
`unix:j*@epsilon3.ibm.com=jbrady`	Associates any Host Based Authentication 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 Host Based Authentication 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 58 shows several examples of how an entry can use the ***** wildcard character when specifying the host identification portion of the security network identifier.

Table 58. 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 Host Based Authentication 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 Host Based Authentication 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 Host Based Authentication 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 59 shows several examples of this.

Table 59. 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 Host Based Authentication 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 Host Based Authentication 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. These are the **<iw>**, **<cluster>** and **<any_cluster>** reserved words.

The **<iw>** reserved word refers to the local node. So, for example, the entry:

```
unix:tardis@<iw>=root
```

will associate the security network identifier *tardis* on the local node with the local user *root*.

The **<cluster>** reserved word refers to any host in the currently active cluster. So, for example, the entry:

```
unix:tardis@<cluster>=root
```

will associate any security network identifier that contains the user name *tardis* and originates from any host in the currently active cluster with the local *root* user. For example, 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. So, for example, the entry:

```
unix:tardis@<any_cluster>=root
```

will associate 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. For example, 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, as described in "Configuring the Host Based Authentication (HBA) mechanism mappings" on page 229. 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.

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 `hba2:`.

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. Refer to "Configuring the Host Based Authentication (HBA) mechanism mappings" on page 229 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. Refer to “Configuring the Host Based Authentication (HBA) mechanism mappings” on page 229 for more information about using the reserved words.

Impacts of alternate authorization mechanisms on identity mapping

As discussed in “Understanding alternate authorization mechanisms” on page 212, 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.

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.

Chapter 8. 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 (as described in “Bringing the peer domain online” on page 29) or when you explicitly create a group using the **mkcomg** command (as described in “Creating a communication group” on page 45).

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 Chapter 9, “The Group Services subsystem,” on page 253. 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 contained in the executable file **/usr/sbin/rsct/bin/hatsd**. This daemon runs on each node in the RSCT peer domain. Note that the operational domain of the Topology Services subsystem is the RSCT peer domain.

When each daemon starts, it first reads its configuration from a file set up by the Startup command (**cthats**). This file is called the machines list file, because it has all 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 Topology Services subsystem directive 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 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 and this daemon is the Group Leader in each one of these singleton groups.

To manage the changes in these groups, Topology Services 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
: This is 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 are 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.

When a network adapter fails or has a problem in one node, this will initially cause incoming heartbeats to be lost. To be able to distinguish a local adapter failure from remote adapter failures, Topology Services will invoke a function which uses *self-death* logic. This self-death logic will attempt to determine whether the adapter is still working. This invokes network diagnosis to determine if the adapter is able to receive data packets from the network. The daemon will try to have data packets sent to the adapter. If it cannot receive any network traffic, the adapter is considered to be down. Group Services is then notified that all adapters in the group are down.

After an adapter that was down recovers, the daemon will eventually find that the adapter is working again, by using a mechanism similar to the self-death logic, and will form a singleton group with it. This should allow the adapter to form a larger group with the other adapters in the network. An *adapter up* notification for the local adapter is sent to the Group Services subsystem.

Pluggable NIMs

Topology Services pluggable NIMs are processes started by the Topology Services daemon 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 coming 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.

Port numbers and sockets

The Topology Services subsystem uses several types of communications:

- UDP port numbers for intracluster communications, that is, communications 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 new peer domain” on page 27).

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 cthatsctrl control command

The Topology Services control command is contained in 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 fact, in an RSCT peer domain, the configuration resource manager controls the Topology Services subsystem, and using this command directly could yield undesirable results. In an RSCT peer domain, you should use this command only if instructed to do so by IBM service.

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 the command to remove the subsystem from the cluster, start the subsystem, stop the subsystem, and build the configuration file for the subsystem.

The cthats startup command

The Topology Services startup command **cthats** is contained in 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. Topology Services is started automatically by the configuration resource manager when you issue the **startrpdomain** or **mkcomg** commands. See “Bringing the peer domain online” on page 29 and “Creating a communication group” on page 45 for more information on the **startrpdomain** and **mkcomg** commands. However if a problem occurs, the users may need to run the **cthatsctrl** command to operate the Topology Services subsystem.

Note: If using RSCT in conjunction with PSSP running a DCE security environment, the Topology Services startup script will run a conversion program that will convert the DCE key into a cluster compatible key. This will allow nodes running Topology Services using cluster security services to coexist with nodes running Topology Services using DCE.

The cthatstune tuning command

The Topology Services tuning command **cthatstune** is contained in the executable file **/usr/sbin/rsct/bin/cthatstune**. The purpose of the **cthatstune** command is to change the Topology Services’ tunable parameters at runtime. When a communication group is created, Topology Services is, under normally operating conditions, configured with the default values for these parameters or values you supply to the **mkcomg** command. These parameters can be modified using the **chcomg** command as described in “Modifying a communication group’s characteristics” on page 42. You can also use the **cthatstune** command to adjust the parameters directly. The **chcomg** and **cthatstune** commands both allow you to change the parameters without restarting the Topology Services subsystem.

For more information about the **cthatstune** command, refer to its online man page. For complete syntax on all RSCT commands, you can also refer to the *Reliable Scalable Cluster Technology for AIX 5L: Technical Reference* or the *Reliable Scalable Cluster Technology for Linux: Technical Reference*.

Files and directories

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 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 topology 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 45 and “Modifying a communication group’s characteristics” on page 42. You can also use the **cthatstune** command (as described in “The cthatstune tuning command” on page 239) 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 245 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 it 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 new peer domain" on page 27 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 is provided to supply additional problem determination information when it is requested by the IBM Support Center. Normally, you should not turn tracing on 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.

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 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 lo0 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 will be 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 given pair of nodes.

Tuning the Topology Services subsystem

The default settings for the frequency and sensitivity tunable attributes discussed in “Configuring Topology Services” on page 243 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 60.

Table 60. 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 61.

Table 61. 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 62.

Table 62. 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 63.

Table 63. 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 42). 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 5L: On AIX 5L 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 5L System Management Guide: Communications and Networks* online manual. Go to http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/ commadmn/tcp_etherchannel.htm for this information:

If the preceding URL no longer points to an active Web page, go to http://www.ibm.com/servers/aix/library/ for AIX documentation.

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 distruption 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 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 this 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. The Topology Services subsystem needs to be refreshed 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.

Chapter 9. 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 will, if necessary, start Group Services. Under normal operating conditions, it will not be necessary for you to directly influence Group Services.

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 Chapter 8, "The Topology Services subsystem," on page 235.

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

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 contained in the executable file **/usr/sbin/rsct/bin/hagsd**. This daemon runs on each node in the peer domain

A GS client communicates with a Group Services daemon that is running on the same node as the GS client. A GS client communicates with the Group Services daemon, through the GSAPI software, using a UNIX domain socket. For HACMP, before a GS client registers with Group Services, it must set the **HA_DOMAIN_NAME** and the **HA_GS_SUBSYS** environment variables to the HACMP cluster name and ″grpsvcs″ respectively. In an RSCT peer domain, the **HA_DOMAIN_NAME** and the **HA_GS_SUBSYS** environment variables *should not* be set.

The Group Services API (GSAPI)

The Group Services Application Programming Interface (GSAPI) is a shared library that a GS 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 64 on page 255 gives the path names for each version of the GSAPI shared library on Linux nodes and AIX nodes.

Table 64. Path names for the GSAPI shared library

	On Linux nodes	On AIX nodes
Non-thread-safe version	/usr/lib/libha_gs.so	/usr/lib/libha_gs.a
Thread-safe version	/usr/lib/libha_gs_r.so	/usr/lib/libha_gs_r.a

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 details on the GSAPI software, see the *Group Services Programming Guide and Reference*.

Allowing 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 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 new peer domain” on page 27).

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 (via 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. In an RSCT peer domain, you should use this command only if instructed to do so by an IBM service representative.

For more information, see "Configuring Group Services" on page 257.

Files and directories used by Group Services

Table 65 summarizes the directories that the Group Services subsystem uses on Linux nodes and AIX nodes.

Table 65. Directories used by the Group Services subsystem

	On Linux nodes	**On AIX nodes**
Lock files	/var/ct/*cluster_name*/lck	/var/ct/*cluster_name*/lck/cthags
Log files	/var/ct/*cluster_name*/log	/var/ct/*cluster_name*/log/cthags
Working directory for the Group Services daemon	/var/ct/*cluster_name*/run	/var/ct/*cluster_name*/run/cthags
Socket files	/var/ct/*cluster_name*/soc	/var/ct/*cluster_name*/soc/cthags

Lock files

On Linux nodes, lock files are located in **/var/ct/***cluster_name***/lck**. On AIX 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 Linux nodes, log files are located in **/var/ct/***cluster_name***/log**. On AIX nodes, log files are located in **/var/ct/***cluster_name***/log/cthags**. The log file directory contains trace output from the Group Services daemon.

On the nodes, the files are called **cthags_***nodenum_instnum***.***cluster_name*, **cthags_***nodenum_instnum***.***cluster_name***.long**, and **cthags.default.***nodenum_instnum*, where:

nodenum is the node number on which the daemon is running.

instnum is the instance number of the daemon.

The Group Services daemon limits the log size to a pre-established number of lines (by default, 5,000 lines). When the limit is reached, the daemon appends the string **.bak** to the name of the current log file and begins a new log. If a **.bak** version already exists, it is removed before the current log is renamed.

Working directory for Group Services daemon

On Linux, the working directory for the Group Services daemon is **/var/ct/***cluster_name***/run**. On AIX nodes, it is **/var/ct/***cluster_name***/run/cthags**. If the Group Services daemon abnormally terminates, the core dump file is placed in this working directory. Whenever the Group Services daemon starts, it renames any core file to **core_***nodenum.instnum*, where *nodenum* is the node number on which the daemon is running and *instnum* is the instance number of the previous instance of the daemon.

Components on which Group Services depends

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 (that is, are running) at any given time. It is often referred to as **heartbeat**. The Topology Services subsystem is SRC-controlled. It is called **cthats**. For more information, see Chapter 8, "The Topology Services subsystem," on page 235.

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 topology 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, you should 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 that 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 simply 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.

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.
2. Based on the input from Topology Services, each daemon finds the lowest-numbered running node in the domain. The daemon compares its own node number to the lowest-numbered node and performs one of the following:
   - If the node the daemon is on is the lowest-numbered 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 lowest-numbered node, it sends nomination messages to the lowest-numbered node periodically, initially every 5 seconds.
3. 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.
4. At this point, the Group Services domain is established, and requests by clients to join or subscribe to groups are processed.

Note that this description 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 domain's GS nameserver. In that case, the GS nameserver waits only for Topology Services to identify the newly running nodes. The GS nameserver will then send the newly running nodes proclaim messages that direct the nodes to nominate it as nameserver. Once those nodes then nominate the GS nameserver, the GS nameserver simply executes one or more insert protocols to insert the newly-running nodes into the domain.

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.

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:

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."

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.

Appendix A. RSCT network considerations

In order for RSCT components to run on and communicate over a network, you should 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.

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, refer to their online man pages.

RSCT port usage

The tables that follow 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 66 on page 264 describes the service definition for the Topology Services subsystem.

Table 66. 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. Please 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 new peer domain" on page 27). The Topology Services port is dynamically added in **/etc/service**, and is only present when the node is online in the peer domain.				

Table 67 describes the service definition for the Group Services subsystem.

Table 67. 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 new peer domain" on page 27). The Group Services port is dynamically added in **/etc/service**, and is only present when the node is online in the peer domain.				

Table 68 describes the service definitions for the Resource Monitoring and Control subsystem.

Table 68. 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
Description: Network services used by RMC for communication between nodes. The UDP port is used for communication between RMC daemons. The TCP port is used for remote client connections to an RMC daemon. The port listed is used by the RMC daemon. The client uses this port to establish a connection to the RMC daemon. To ensure full RMC functionality, both the UDP port and the TCP port must be enabled. The client half of the connection uses an ephemeral port.				

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.

The term *peer domain* is defined as a set of nodes which have a consistent knowledge of the existence of each other and of the resources shared between 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. Refer to Chapter 3, “Creating and administering an RSCT peer domain,” on page 17 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 automatically created in a POWER4™ pSeries system or a System p5™ system, 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, the IBM Cluster Systems Management (CSM) products each create 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 firewalls on the RMC subsystem and any RMC client application.

Port usage

The RMC daemon uses the TCP port 657 to accept requests from RMC client applications, such as CSM and the various RMC commands that are executing remotely from the node upon which the RMC daemon is running. The RMC commands are documented in the RSCT Technical Reference. 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 or if the node is an operating system image in a POWER4 pSeries system or a System p5 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 266, 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, for both the TCP RMC protocols and the UDP RMC protocols, message authentication is employed. 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 found in RSCT 2.3.1.0 or greater on AIX 5.2 and in RSCT 2.4.0.0 or greater on AIX 5.3 and Linux. UDP message authentication is found in RSCT 2.3.5.0 or greater on AIX 5.2 and in RSCT 2.4.1.0 or greater on AIX 5.3 and Linux.

Management domain configuration

When an operating system image is booted on a POWER4 pSeries system or a System p5 system, the RMC subsystem is started and then the IBM.CSMAgentRM resource manager is started. This resource manager implements the necessary function to automatically add the operating system image, as a managed node, to the management domains created automatically on each HMC. This resource manager initiates a series of sessions to the RMC daemon on each HMC, as described above:

- A session to each configured IP address of each HMC to validate the IP address. No commands are sent.

- 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. Once it is a managed node, the RMC daemon begins communication with the RMC daemon on each HMC, as described above. Periodically, the IBM.CMSAgentRM resource manager 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.

When the **updatenode** command is executed on a CSM Management Server, the **dsh** command is used to execute the **mgmtsvr** command on each node that is specified to the **updatenode** command. The **mgmtsvr** command then triggers the IBM.CSMAgentRM on the node to initiate a series of sessions to the RMC daemon on the Management Server:

- A session to the MS. A command is sent to determine the code level on the MS of the IBM.DMSRM resource manager.
- A session to the MS. 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. Once it is a managed node, the RMC daemon begins communication with the RMC daemon on the Management Server, as described above.

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
```

Output will be similar to:

```
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
```

Output will be similar to:

```
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.

Appendix B. Product-related information

Reliable Scalable Cluster Technology (RSCT) is a component of the following licensed programs:
- AIX 5L
- Cluster Systems Management (CSM) for Linux
- System Automation for Multiplatforms

RSCT version

This edition applies to RSCT version:
- | 2.3.10.0 for AIX 5.2
- | 2.4.6.0 for AIX 5.3 and Linux

To find out which version of RSCT is running on a particular AIX node, enter:

```
lslpp -L rsct.basic.rte
```

To find out which version of RSCT is running on a particular Linux node, enter:

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

ISO 9000

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

Product-related feedback

To contact the IBM cluster development organization, send your comments by e-mail to:

cluster@us.ibm.com

Appendix C. Accessibility features for RSCT

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Reliable Scalable Cluster Technology (RSCT). These features support:
- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Customization of display attributes, such as color, contrast, and font size

Note: The IBM Cluster Information Center and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Related accessibility information

Assistive technology products, such as screen readers, function with user interfaces. Consult the assistive technology documentation for specific information when using such products to access interfaces.

IBM and accessibility

See the *IBM Accessibility Center* at http://www.ibm.com/able for more information about the commitment that IBM has to accessibility.

Notices

Trademarks

Glossary

access control. The process of limiting access to system objects and resources to authorized principals.

access control list. A list of principals and the type of access allowed to each.

ACL. See *access control list.*

action. The part of the event response resource that contains a command and other information about the command.

attribute. Attributes are either persistent or dynamic. A resource class is defined by a set of persistent and dynamic attributes. A resource is also defined by a set of persistent and dynamic attributes. Persistent attributes define the configuration of the resource class and resource. Dynamic attributes define a state or a performance-related aspect of the resource class and resource. In the same resource class or resource, a given attribute name can be specified as either persistent or dynamic, but not both.

AIX. Advanced Interactive Executive. See *AIX operating system.*

AIX operating system. IBM's implementation of the UNIX operating system.

authentication. The process of validating the identity of an entity, generally based on user name and password. However, it does not address the access rights of that entity. Thus, it simply makes sure a user is who he or she claims to be.

authorization. The process of granting or denying access to an entity to system objects or resources, based on the entity's identity.

checksum. A count of the number of bits in a transmission unit that is included with the unit so that the receiver can check to see whether the same number of bits arrived. If the counts match, it's assumed that the complete transmission was received. TCP and UDP communication layers provide a checksum count and verification as one of their services.

client. Client applications are the ordinary user interface programs that are invoked by users or routines provided by trusted services for other components to use. The client has no network identity of its own: it assumes the identity of the invoking user or of the process where it is called, who must have previously obtained network credentials.

cluster. A group of servers and other resources that act like a single system and enable high availability and, in some cases, load balancing and parallel processing.

clustering. The use of multiple computers (such as UNIX workstations, for example), multiple storage devices, and redundant interconnections to form what appears to users as a single highly-available system. Clustering can be used for load balancing, for high availability, and as a relatively low-cost form of parallel processing for scientific and other applications that lend themselves to parallel operations.

cluster security services. A component of RSCT that is used by RSCT applications and other RSCT components to perform authentication within both management domains and peer domains.

condition. A state of a resource as defined by the event response resource manager (ERRM) that is of interest to a client. It is defined by means of a logical expression called an event expression. Conditions apply to resource classes unless a specific resource is designated.

condition/response association. A link between a condition and a response.

CSM. Clusters Systems Management.

datagram. Synonymous with *UDP packet.*

DNS. See *domain name system.*

domain. (1) A set of network resources (such as applications and printers, for example) for a group of users. A user logs in to the domain to gain access to the resources, which could be located on a number of different servers in the network. (2) A group of server and client machines that exist in the same security structure. (3) A group of computers and devices on a network that are administered as a unit with common rules and procedures. Within the Internet, a domain is defined by its IP address. All devices that share a common part of the IP address are said to be in the same domain.

domain name. A meaningful and easy-to-remember ″handle″ for an Internet address.

domain name system. The service through which domain names are located and translated into IP addresses.

event. Occurs when the event expression of a condition evaluates to True. An evaluation occurs each time an instance of a dynamic attribute is observed.

event expression. A definition of the specific state when an event is true.

event response. One or more actions as defined by the event response resource manager (ERRM) that take place in response to an event or a rearm event.

failover. A backup operation that automatically switches to another adapter if one adapter fails. Failover is an important fault-tolerance function of mission-critical systems that rely on constant accessibility. Automatically and transparently to the user, failover redirects requests from the failed adapter to another adapter that mimics the operations of the failed adapter.

FFDC. See *first failure data capture.*

first failure data capture. Provides a way to track problems back to their origin even though the source problem may have occurred in other layers or subsystems than the layer or subsystem with which the end user is interacting. FFDC provides a correlator called an **ffdc_id** for any error that it writes to the AIX error log. This correlator can be used to link related events together to form a chain.

FIFO. First in first out, usually referring to buffers.

High Performance Switch. The switch that works in conjunction with a specific family of IBM pSystem servers.

HPS. See *High Performance Switch.*

Internet Protocol. The method by which data is sent from one computer to another on the Internet.

IP. See *Internet Protocol.*

IP address. A 32-bit (in IP Version 4) or 128-bit (in IP Version 6) number identifying each sender or receiver of information that is sent in packets across the Internet.

LAPI. See *low-level application programming interface.*

Linux. A freeware clone of UNIX for 386-based personal computers (PCs). Linux consists of the **linux** kernel (core operating system), originally written by Linus Torvalds, along with utility programs developed by the Free Software Foundation and by others.

LoadLeveler®. A job management system that works with POE to let users run jobs and match processing needs with system resources, in order to make better use of the system.

low-level application programming interface. A low-overhead message-passing protocol that uses a one-sided communication model and active message paradigm to transfer data among tasks. See also *RSCT LAPI.* Contrast with *PSSP LAPI.*

logical unit number. A unique identifier used on a SCSI bus that enables it to differentiate between up to eight separate devices (each of which is a logical unit). Each LUN is a unique number that identifies a specific logical unit, which may be an end user, a file, or an application program.

LUN. See *logical unit number.*

management domain. A set of nodes configured for manageability by the Clusters Systems Management (CSM) licensed program. Such a domain 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; they know nothing of each other. Contrast with *peer domain.*

Message Passing Interface. A standardized API for implementing the message-passing model.

MPI. See *Message Passing Interface.*

mutex. See *mutual exclusion object.*

mutual exclusion object. A program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously. When a program is started, a mutual exclusion object is created with a unique name. After this stage, any thread that needs the resource must lock the mutual exclusion object from other threads while it is using the resource. The mutual exclusion object is set to unlock when the data is no longer needed or the routine is finished.

network credentials. These represent the data specific to each underlying security mechanism.

OSI. Operating system image.

PAC. See *privileged attribute certificate.*

packet. The unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network.

| **Parallel Environment.** An IBM licensed program that
| is an execution and development environment for
| parallel C, C++, and FORTRAN programs. PE also
| includes tools for debugging, profiling, and tuning
| parallel programs.

parallel operating environment. An execution environment that smooths the differences between serial and parallel execution. It lets you submit and manage parallel jobs.

Parallel System Support Programs. The IBM Parallel System Support Programs for AIX 5L licensed program is system administration software for the IBM RS/6000® SP system.

PE. See *Parallel Environment.*

peer domain. A set of nodes configured for high availability by the configuration resource manager. Such a domain has no distinguished or master node. All nodes are aware of all other nodes, and administrative commands can be issued from any node in the domain. All nodes also have a consistent view of the domain membership. Contrast with *management domain.*

POE. See *parallel operating environment.*

port. A ″logical connection place″. Using TCP/IP, the way a client program specifies a particular server program on a computer in a network.

principal. A user, an instance of the server, or an instance of a trusted client whose identity is to be authenticated.

privileged attribute certificate. Contains such information as the client's name and the groups to which it belongs. Its format is dependent on the underlying security mechanism.

protocol. The set of rules that endpoints in a telecommunication connection use when they communicate.

PSSP. See *Parallel System Support Programs.*

PSSP LAPI. The version of LAPI that supports the SP Switch2.

rearm event. Occurs when the rearm expression for a condition evaluates to True.

rearm expression. An expression that generates an event which alternates with an original event in the following way: the event expression is used until it is true; then, the rearm expression is used until it is true; then, the event expression is used. The rearm expression is commonly the inverse of the event expression. It can also be used with the event expression to define an upper and lower boundary for a condition of interest.

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

resource. An entity in the system that provides a set of services. Examples of hardware entities are processors, disk drives, memory, and adapters. Examples of software entities are database applications, processes, and file systems. Each resource in the system has one or more attributes that define the state of the resource.

resource class. A broad category of system resource, for example: node, file system, adapter. Each resource class has a container that holds the functions, information, dynamic attributes, and conditions that apply to that resource class. For example, the **/tmp space used** condition applies to a file system resource class.

resource manager. A process that maps resource and resource-class abstractions into calls and commands for one or more specific types of resources. A resource manager can be a standalone daemon, or it can be integrated into an application or a subsystem directly.

RSCT. See *Reliable Scalable Cluster Technology.*

| **RSCT LAPI for AIX 5L.** The version of LAPI that
| supports the IBM eServer High Performance Switch
| (HPS). See also *low-level application programming*
| *interface.*

| **RSCT LAPI for Linux.** The version of LAPI that
| supports the InfiniBand switch. See also *low-level*
| *application programming interface.*

RSCT peer domain. See *peer domain.*

SCSI. See *Small System Computer Interface.*

Small System Computer Interface. A parallel interface that can have up to eight devices all attached through a single cable; the cable and the host (computer) adapter make up the SCSI bus. The bus allows the interchange of information between devices independently of the host. In the SCSI program, each device is assigned a unique number, which is either a number between 0 and 7 for an 8-bit (narrow) bus, or between 8 and 16 for a 16-bit (wide) bus. The devices that request input/output (I/O) operations are initiators and the devices that perform these operations are targets. Each target has the capacity to connect up to eight additional devices through its own controller; these devices are the logical units, each of which is assigned a unique number for identification to the SCSI controller for command processing.

SD. Structured data.

security context token. A pointer to an opaque data structure called the context token descriptor. The context token is associated with a connection between a client and the server.

security services token. A pointer to an opaque descriptor called the security token descriptor. It keeps track of the mechanism-independent information and state.

servers. Server programs are usually daemons or other applications running in the background without a user's inherited credentials. A server must acquire its own network identity to get to access other trusted services.

SP Switch2. The switch that works in conjunction with IBM RS/6000 SP systems.

| **standalone system.** A system on which you are using
| RSCT LAPI for AIX 5L or LAPI for Linux that is not
| running PE.

striping. The distribution of message data across multiple communication adapters in order to increase bandwidth.

TCP. See *Transmission Control Protocol.*

Transmission Control Protocol. One of the core Internet protocols. TCP ports are 16-bit entities, so a maximum of 65535 different endpoints are possible within a single IP address.

UDP. See *User Datagram Protocol.*

User Datagram Protocol. One of the core Internet protocols. UDP is a layer 4 protocol (Transport layer of the OSI model) within the Internet protocol suite. It provides a mechanism to identify different endpoints on a single host by using ports. UDP deals with single-packet delivery that is provided by the underlying IP. As a stateless protocol, it is often used in applications where data must arrive quickly. This smaller feature set provides quicker data transmittal and lower total overhead. UDP packets (or *datagrams*) contain, in addition to the lower-level headers, a UDP header, which consists of the packet length, source and destination ports, and a checksum. UDP ports are 16-bit entities, so a maximum of 65535 different endpoints are possible within a single IP address.

Index

Readers' Comments — We'd Like to Hear from You

Reliable Scalable Cluster Technology
Administration Guide

Publication No. SA22-7889-11

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SA22-7889-11

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie NY 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

SA22-7889-11

Cut or Fold
Along Line

Program Number: 5765-E62, 5765-G03, 5765-E88, 5765-G16, 5724-M00

SA22-7889-11