IBM

System i

Systems management Clusters

Version 5 Release 4

System i

Systems management
Clusters

Version 5 Release 4

Note

Before using this information and the product it supports, read the information in "Notices," on page 171.

Seventh Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Clusters

Clusters let you efficiently group your System i™ products together to set up an environment that provides availability that approaches 100 percent for your critical applications, devices, and data.

Clusters also provide simplified systems management and increased scalability to seamlessly add new components as your business grows.

By using the code examples, you agree to the terms of the Code license and disclaimer information.

What's new for V5R4

Take a look at what is new for this release.

Cluster administrative domain support

A *cluster administrative domain* monitors and synchronizes changes to selected resources within a cluster. Cluster administrative domain provides easier management and synchronization of attributes for resources that are shared within a cluster, such as environment variables or user profiles. For more information about cluster administrative domain, see these topics:

- "Cluster administrative domain" on page 9
- "Planning for a cluster administrative domain" on page 109
- "Cluster administrative domain checklist" on page 110
- "Creating a cluster administrative domain" on page 121

Peer cluster resource group (CRG) support

All CRG interfaces have been enhanced to support peer cluster resource group (CRG). A *peer cluster resource group (CRG)* is a non-switchable CRG in which each node in the recovery domain has an equal role in the recovery of the resources associated with the peer CRG. For more information, see the following topics:

- Cluster resource group
- "Creating a CRG" on page 116
- "Starting a CRG" on page 117

Cluster enhancements

Several enhancements have been made to improve powering down operations and problem resolution within a clustered environment. These improvements include:

- A systematic approach to end clustering on a cluster node when all active subsystems are ended, or when the system is ended or powered down. See "How a system event affects a cluster" on page 121 for details.
- The capability to configure a new application CRG with an active takeover IP address. See "Creating an application CRG with an active takeover IP address" on page 117 for more information.
- The capability to troubleshoot cluster problems by viewing an entire cluster and its associated CRGs from an active node. See "Gathering recovery information for a cluster" on page 141 for more details.
- New information about debug tools and the results they generate has been added. You can use these tools and their results to determine the resolution of a problem within a cluster. See the following topics for more details:
  - "Investigating a problem with Dump Cluster Trace (DMPCLUTRC) command" on page 142

- "Investigating a problem with CLUSTERINFO macro" on page 146

What's new as of 27 February 2006

Additional information was added to help users perform upgrades in a clustered environment. See "Cluster version" on page 20 for more details.

What's new as of 25 July 2006

Information about working with the administrative domain has been modified. In addition, these new topics have been added:

- "Adding a node to the cluster administrative domain" on page 124
- "Starting and ending the peer CRG" on page 125
- "Starting or ending a cluster administrative domain node" on page 125
- "Partitioned cluster administrative domains" on page 126
- "Attributes that can be monitored" on page 12

What's new as of 31 October 2006

Suppose an unsuccessful switchover occurs for a device CRG. After switching back all the devices, if all of the devices were varied-on successfully on the original primary node, clustering calls the exit program on the original primary node with an action code of Start. See "Calling a cluster resource group exit program" on page 41.

What's new as of 31 December 2006

Information about working with the administrative domain and monitored resource entries (MREs) has been modified. See the following topics:

- "Starting and ending the peer CRG" on page 125
- "Adding and removing monitored resource entries" on page 122
- "Cluster administrative domain checklist" on page 110

How to see what's new or changed

To help you see where technical changes have been made, this information uses:

- The » image to mark where new or changed information begins.
- The « image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to Users.

Printable PDF

Use this to view and print a PDF of this information.

To view or download the PDF version of this document, select Clusters (about 938 KB).

Redbooks

- Clustering and IASPs for Higher Availability (about 6.4 MB) This redbook presents an overview of cluster and switched disk technology available for System i products.

- iSeries™ Independent ASPs: A Guide to Moving Applications to IASPs (about 3.4 MB) This redbook presents a step-by-step approach to independent ASPs on System i products.

- Roadmap to Availability on the iSeries 400 (about 626 KB) This redpaper presents a step-by-step approach to independent ASPs on System i products.

Web sites

- High Availability and Clusters (www.ibm.com/servers/eserver/iseries/ha)
  IBM site for High Availability and Clusters

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As** if you are using Internet Explorer. Click **Save Link As** if you are using Netscape Communicator.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

You need Adobe Acrobat Reader to view or print these PDFs. You can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html).

Cluster concepts

Get a complete understanding of how clusters work. Read about the benefits of clusters and how they can be important to you, as well as information on important clustering concepts and how they fit together.

A System i cluster is a collection or group of one or more systems or logical partitions that work together as a single system. Systems in a cluster, called cluster nodes, work cooperatively to provide a single computing solution. System i clustering supports up to 128 nodes in a cluster. This allows you to efficiently group your System i products together to set up an environment that provides availability that approaches 100 percent for your critical applications and your critical data. This helps ensure that your critical systems and applications are available 24 hours a day, seven days a week. Clusters also provide simplified systems management and increased scalability to seamlessly add new components as your business grows.

Benefits of clusters

Clusters provide a solution if your business demands operational systems 24 hours a day, seven days a week.

By using clustering, you can greatly reduce the number and duration of unplanned outages and the duration of planned outages, thereby ensuring that your systems, data, and applications are continuously available.

The major benefits that clusters can offer your business are:

Continuous availability
Clusters ensure that your systems, data and applications remain continuously available.

Simplified administration
You can manage a group of systems as a single system or single database, without having to sign on to individual systems. You can use a cluster administrative domain to more easily manage resources that are shared within a cluster.

Increased scalability
Seamlessly add new components as your business growth requires.

Related concepts

"Failover" on page 26
Failover occurs when a server in a cluster automatically switches over to one or more backup servers in the event of a system failure.

Related tasks

"Switchover" on page 29
Switchover happens when you manually switch access to a resource from one server to another.

How a cluster works

The cluster infrastructure provided as a part of i5/OS®, called cluster resource services, provides resiliency for your critical resources. These resources can include data, applications, devices, and other resources accessed by multiple clients.

| If a system outage or a site loss occurs, the functions that are provided on a system within a cluster can
| be accessed through other systems that have been defined in the cluster. There are two models in which
| this data can be accessed: primary-backup model and peer model. For more details on cluster resource
| groups (CRGs) that you can create based on these models, see Cluster resource group.

Related concepts

"Failover" on page 26
Failover occurs when a server in a cluster automatically switches over to one or more backup servers in the event of a system failure.

"Replication" on page 34
Replication makes a copy of something in real time. It means copying objects from one node in a cluster to one or more other nodes in the cluster.

"Resilient devices" on page 24
Resilient devices are physical resources, represented by a configuration object, such as a device description, that are accessible from more than one node in a cluster.

"Resilient data" on page 24
Resilient data is data that is replicated (copied) on more than one node in a cluster.

"Rejoin" on page 30
Rejoin means to become an active member of a cluster after having been a nonparticipating member.

"Comparison of logical replication, switched disks, and cross-site mirroring" on page 98
This topic provides an overview of different data resilience technologies that can be used with clusters to enhance high availability.

Related tasks

"Switchover" on page 29
Switchover happens when you manually switch access to a resource from one server to another.

Cluster basics

Understand the basic clustering concepts before you begin to design and customize a cluster to satisfy your needs.

| There are two basic concepts related to a cluster: cluster nodes and cluster resource groups. A *cluster node*
| is either a System i product or a logical partition that is a member of the cluster. When you create a
| cluster, you specify the systems or logical partitions that you want to include in the cluster as nodes. A
| *cluster resource group (CRG)* serves as the control object for a collection of resilient resources. A CRG may
| contain a subset or all of nodes within the cluster. A System i cluster supports four types of CRGs:
| application, data, device, and peer. Within these types of CRGs there are two common elements: a
| recovery domain and an exit program.

A *recovery domain* defines the role of each node in the CRG. When you create a CRG in a cluster, the CRG object is created on all nodes specified to be included in the recovery domain. However, a single system image of the CRG object, which you can access from any active node in the CRG's recovery domain, is provided. That is, any changes made to the CRG will be made on all nodes in the recovery domain.

An *exit program* is called during cluster-related events for the CRG. One such event is moving an access point from one node to another node.

There are two models of CRGs that can be created in a cluster: primary-backup model and peer model. In the primary-backup model, the nodes in the recovery domain of the CRG can be defined as follows:

- The *primary node* is the cluster node that is the primary point of access for the resilient cluster resource.
- A *backup node* is a cluster node that will take over the role of primary access if the present primary node fails or a manual switchover is initiated.
- A *replicate node* is a cluster node that has copies of cluster resources, but is unable to assume the role of primary or backup.

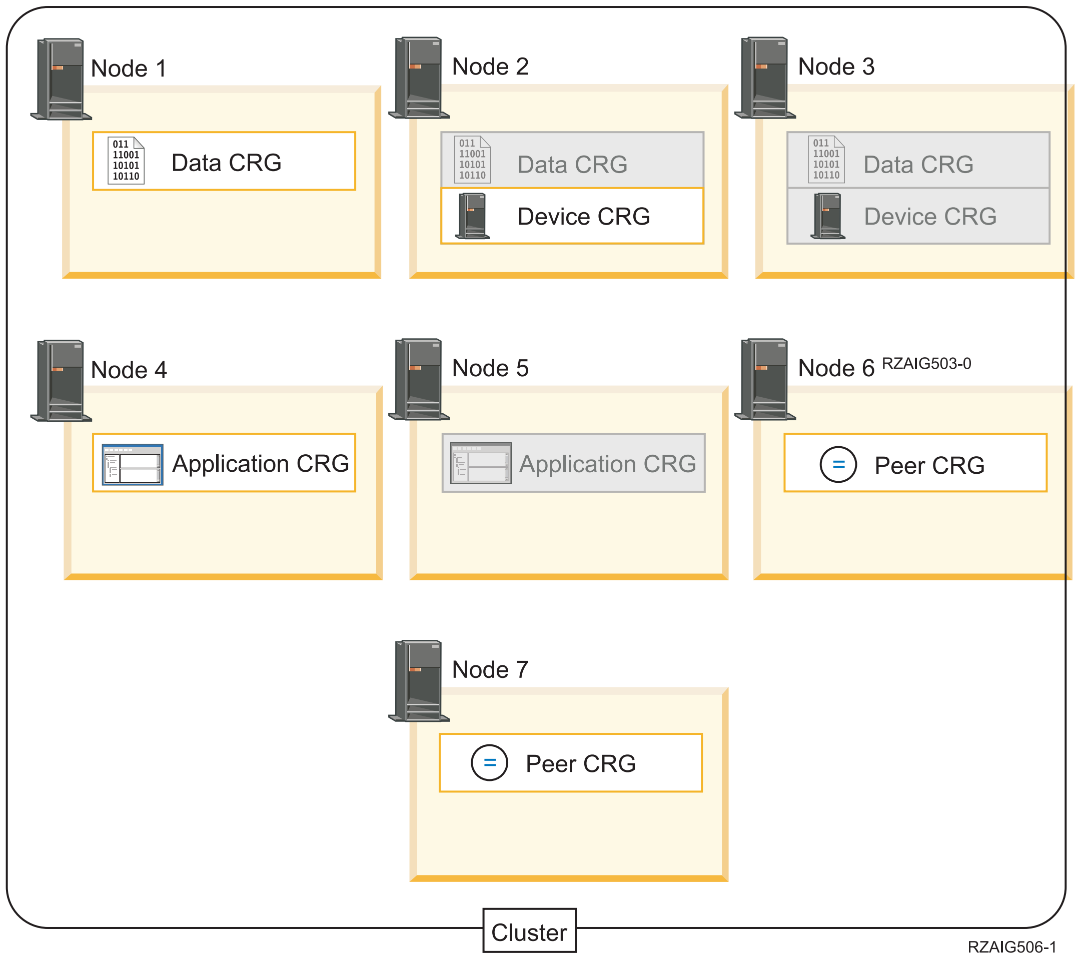
In a peer model, the recovery domain of a peer CRG defines a peer relationship between nodes. The nodes in the recovery domain of the peer CRG can be defined as follows:

- A *peer node* is a cluster node that can be an active access point for cluster resources.
- A *replicate node* is a cluster node that has copies of cluster resources. Nodes defined as replicate in a peer CRG represent the inactive access point for cluster resources.

With a peer CRG, the nodes in the recovery domain are equivalent with respect to the role the nodes plays in recovery. Because each node in this peer CRG has essentially the same role, the concepts of a failover and switchover do not apply. The nodes have a peer relationship, and when one of the nodes fails, others peer nodes will continue operating.

You can also create a cluster administrative domain which is represented by a peer CRG. The nodes in a cluster administrative domain are all peer nodes in the CRG's recovery domain. There are no replicate nodes.

In the example below, one CRG of each type is present:



Data CRG
The data CRG is present on Node 1, Node 2 and Node 3. This means that the recovery domain for the data CRG has specified a role for Node 1 (primary), Node 2 (first backup) and Node 3 (second backup). In the example, Node 1 is currently serving as the primary point of access. Node 2 is defined as the first backup in the recovery domain. This means that Node 2 contains a copy of the resource which is kept current through logical replication. Should a failover or switchover occur, Node 2 becomes the primary point of access.

Application CRG
The application CRG is present on Node 4 and Node 5. This means that the recovery domain for the application CRG has specified Node 4 and Node 5. In the example, Node 4 is currently serving as the primary point of access. Should a failover or switchover occur, Node 5 becomes the primary point of access for the application. Requires a takeover IP address.

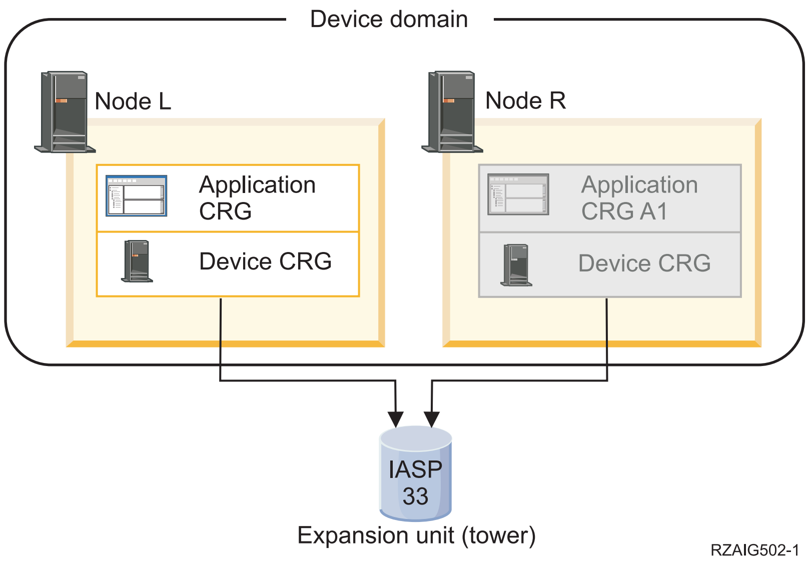
| **Peer CRG**
| The peer CRG is present on Node 6 and Node 7. This means that the recovery domain for the
| peer CRG has specified Node 6 and Node 7. In this example, nodes 6 and 7 can be either peer or
| replicate nodes. If this is a cluster administrative domain that is represented by peer CRG,
| resources that are monitored by the cluster administrative domain will have any changes
| synchronized across the domain represented by node 6 and node 7, regardless on which node the
| change originated.

Device CRG
The device CRG is present on Node 2 and Node 3. This means that the recovery domain for the device CRG has specified Node 2 and Node 3. In the example, Node 2 is currently serving as the

primary point of access. This means that the resilient device owned by the device CRG can currently be accessed from Node 2. Should a failover or switchover occur, Node 3 becomes the primary point of access for the device.

A device CRG requires a resilient device called an independent disk pool (also called an independent auxiliary storage pool or independent ASP) to be configured on an external device, an expansion unit (tower) or IOP in a logical partition.

The nodes in the recovery domain of a device CRG must also be members of the same device domain. The example below illustrates a device CRG with Node L and Node R in its recovery domain. Both nodes are also members of the same device domain.



Related concepts

"Cluster node"
A *cluster node* is a System i product or logical partition that is a member of a cluster.

"Cluster resource group" on page 8
A *cluster resource group (CRG)* is an i5/OS system object that is a set or grouping of cluster resources that are used to manage events that occur in a clustered environment. The cluster resource group describes a recovery domain and supplies the name of the cluster resource group exit program which gets called when certain cluster events occur.

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

"Cluster resource group exit programs" on page 17
A *cluster resource group exit program* is called after a cluster-related event for a CRG occurs.

Independent disk pools

"Device domains" on page 25
A *device domain* is a subset of nodes in a cluster that share device resources. More specifically, nodes in a device domain can participate in a switching action for some collection of resilient device resources.

The elements of a cluster

A System i *cluster* is a collection of one or more systems or partitions that work together as a single system. Use this information to understand the elements and their relationship to each other.

Cluster node

A *cluster node* is a System i product or logical partition that is a member of a cluster.

Each cluster node is identified by an 8-character cluster node name that is associated with one or more IP addresses that represent a system. When configuring a cluster, you can use any name that you want for a node in the cluster. However, it is recommended that the node name be the same as the host name or the system name.

Cluster communications makes use of the TCP/IP protocol suite to provide the communications paths between cluster services on each node in the cluster. The set of cluster nodes that are configured as part of the cluster is referred to as the cluster membership list.

Cluster resource group

A *cluster resource group (CRG)* is an i5/OS system object that is a set or grouping of cluster resources that are used to manage events that occur in a clustered environment. The cluster resource group describes a recovery domain and supplies the name of the cluster resource group exit program which gets called when certain cluster events occur.

| Clusters provides two choices in defining the relationships between nodes within a cluster:
| primary-backup model and peer model. Each of these models can be used together or separately
| depending on the needs of your environment.

Primary-backup model

| All cluster resource groups of this category define nodes in the recovery domain with specific roles: either
| primary, backup or replicate. The primary and backup nodes are available access points for cluster
| resources. However only one node will be the active access point at a given point in time. This will be
| the primary node. Replicate nodes are not available to be an access point. This can be changed by
| assigning the replicate node a role of backup. Primary-backup model cluster resource groups are defined
| as data resilient, application resilient, or device resilient. Data resiliency enables multiple copies of data to
| be maintained on more than one node in a cluster and enables the point of access to be changed to a
| backup node. Application resiliency enables an application program to be restarted on either the same
| node or a different node in the cluster. Device resiliency enables a device resource to be moved
| (switched) to a backup node.

Each data and application cluster resource group has a cluster resource group exit program associated with it. The exit program is optional for resilient device cluster resource groups.

In iSeries Navigator, cluster resource groups are referred to differently.
| • A device CRG is referred to as a **switchable device**.
| • An application CRG is referred to as a **switchable application**.
| • A data CRG is referred to as a **switchable data group**.

Peer model

| All cluster resource groups of this category define nodes in the recovery domain with a role of peer or
| replicate. The peer nodes are available to be the access point for the cluster resource group. All nodes
| defined as peer will be the access point when the cluster resource group is started. Replicate nodes are
| not available to be an access point. This can be changed by assigning the replicate node a role of peer.
| Within a peer CRG, each node contains replicated data that exists on each of the nodes. When a node
| fails in a peer CRG, the failure point is communicated to other nodes in the cluster, and those nodes
| continue the operation from the point of failure.

A cluster administrative domain is represented by a peer CRG with a recovery domain made up of only peer nodes.

Related concepts

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

"Cluster resource group exit programs" on page 17
A *cluster resource group exit program* is called after a cluster-related event for a CRG occurs.

Managing the processing of cluster resource groups

When a node failure occurs, a failover will occur. When sequencing of CRGs occurs, all device CRGs fail over first, then all data CRGs and lastly application CRGs. For peer CRG, there is no sequence order, but each node is notified when a failure occurs.

You can verify that the cluster resource group has completed failover or switchover by checking the status of the CRG.

You can also use blocking to hold the application until the data is available to be processed. While the data-resilient cluster resource groups are being processed, you may want to block access to the data represented by the data CRG. You can block access by using the Block EDRS Access (QxdaBlockEDRS) API and the Check EDRS Block Status (QxdaCheckEDRSBlock) API. If a failover or switchover should occur, you can block and unblock access from within the cluster resource group exit program using these APIs.

Related concepts

"Failover" on page 26
Failover occurs when a server in a cluster automatically switches over to one or more backup servers in the event of a system failure.

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

"Cluster resource group exit programs" on page 17
A *cluster resource group exit program* is called after a cluster-related event for a CRG occurs.

Related tasks

"Switchover" on page 29
Switchover happens when you manually switch access to a resource from one server to another.

Cluster administrative domain

A *cluster administrative domain* is used to maintain a consistent operational environment across a subset of cluster nodes to ensure that a highly available application will behave as expected as it is switched to or failed over to backup nodes.

In a high availability (HA) environment, applications and application data reside on multiple systems so that they are available, even when there is a failure on one of the systems. There are often configuration parameters or data associated with applications and application data. Examples of configuration data include user profiles needed to access the application and its data, or system environment variables that control the behavior of the application. The set of configuration parameters is known collectively as the operational environment for the application. The operational environment for an HA application needs to be the same on every system where the application can run, or where the application data resides. When a change is made to one or more configuration parameters on one system, the same change needs to be made on all systems.

A cluster administrative domain provides the capability to identify resources which need to be maintained consistently across the systems in an HA environment. It then monitors for changes to these resources and synchronizes the changes across the active domain to maintain the consistency of the operational environment.

A cluster administrative domain is represented by a peer CRG. When a cluster administrative domain is created, the peer CRG is created by the system. The name of the cluster administrative domain becomes the name of the peer CRG. Membership in the cluster administrative domain can be changed by adding and removing nodes to the recovery domain of the peer CRG. The nodes which make up the cluster administrative domain are defined by the recovery domain of the peer CRG. All of the nodes are peer nodes. Replicate nodes are not allowed in a cluster administrative domain. A cluster node can only be defined in one cluster administrative domain within the cluster.

For more information on associated cluster administrative domain tasks, see the following topics:

1. "Planning for a cluster administrative domain" on page 109
2. "Cluster administrative domain checklist" on page 110
3. "Creating a cluster administrative domain" on page 121
4. "Adding and removing monitored resource entries" on page 122
5. "Managing a cluster administrative domain" on page 123

You can manage a cluster administrative domain using APIs, CL commands and iSeries Navigator.

Once the cluster administrative domain is created, normal CRG functions are used to manage the cluster administrative domain. See "Managing a cluster administrative domain" on page 123 for details.

If the cluster administrative domain is deleted, all monitored resource entries (MREs) that are defined in the cluster administrative domain are removed from every node in the domain; however the actual resource is not removed from the system. See Monitored resources for details.

Related concepts

"Resilient environments" on page 24
Applications running in a cluster need consistent environments to operate properly. Environment resiliency allows changes to objects and attributes to be automatically propagated throughout a cluster.

"Cluster basics" on page 4
Understand the basic clustering concepts before you begin to design and customize a cluster to satisfy your needs.

Related tasks

"Creating a cluster administrative domain" on page 121
A cluster administrative domain can be created by using iSeries Navigator or the Create Cluster Administrative Domain (CRTADMDMN) command.

Monitored resources

Monitored resources are types of system resources that can be managed by a cluster administrative domain. These resources are represented in the cluster administrative domain as *monitored resource entries (MREs)*.

Resources which are synchronized by the cluster administrative domain are represented by MREs. Once an MRE is added to the cluster administrative domain, changes made to the resource on any node in the cluster administrative domain will be propagated to all nodes in the active domain. You can use iSeries Navigator to add and remove MREs, as well as for monitoring the MREs that you have added. See Add and Remove monitored resource entries and Manage a cluster administrative domain for instructions. There are also Integrated Operating Environments APIs for managing MREs in a cluster administrative domain:

- Add Monitored Resource Entry (QfpadAddMonitoredResourceEntry) API
- Remove Monitored Resource Entry (QfpadRmvMonitoredResourceEntry) API
- Retrieve Monitored Resource Information (QfpadRtvMonitoredResourceInfo) API

A monitored resource entry (MRE) can be added to the cluster administrative domain for the following types of resources. Use the links to find out about specific attributes that can be monitored for each resource type.

- Classes
- Independent disk pools device descriptions
- Job descriptions
- Network attributes
- System environment variables
- System values
- TCP/IP attributes
- User profiles

An MRE can only be added to the cluster administrative domain if all nodes in the domain are active and participating in the group. An MRE cannot be added if the cluster administrative domain is partitioned. You can determine the status of the cluster administrative domain and the status of the nodes in the domain by using iSeries Navigator, the DSPCRGINF CL command or the QcstListCrgInfo API.

In order for an MRE to be added, the resource must exist on the node from which the MRE is added. If the resource does not exist on another node in the administrative domain, it will be created. If a node is subsequently added to the cluster administrative domain and a monitored resource does not exist on that node, it will be created.

Once the MRE is added, changes to the resource represented by the MRE are propagated to all active nodes in the domain when the peer CRG is started.

Once an MRE is added to the cluster administrative domain, the resource is monitored for changes on all administrative domain nodes so that the values of the resource attributes can be synchronized across the nodes in the cluster administrative domain. The synchronization behavior is dependent on a number of factors:

- Status of the cluster
- Status of the cluster administrative domain, i.e. the status of the associated Peer CRG
- Status of the node
- Particular actions on the resource

There are several values that determine the status of a monitored resource across the cluster:

- Global Value: The value for each monitored attribute that a resource is expected to have on all administrative domain nodes. The global value is the same on all active nodes and represents the last change that was synchronized in the domain.
- Global Status: An indication of whether the resources are fully synchronized. If the global status is *consistent*, the resource is synchronized to the global value. A global status of *inconsistent* means that the monitored resource is not synchronized to the global value on one or more active nodes in the administrative domain.
- The actual value of the resource attributes on any particular node in the administrative domain: If the resource attributes are set to the global value, the resource will have a global status of *consistent*. If the actual values of the resource attributes are not the same as the global values, the global status is *inconsistent*.

When the global status is inconsistent, the administrator must determine the cause of the failure and correct it. The cluster administrative domain will attempt to resynchronize the resource the next time it is updated, probably when the administrator changes the resource as a result of fixing the problem which caused an update to fail, or when the CRG is restarted.

Changes made to monitored resources when the cluster administrative domain is inactive make them inconsistent. Therefore, when the cluster administrative domain is ended, the global status for all MREs is set to *inconsistent*.

In the normal operational environment, the cluster is active, the cluster administrative domain is active, and all of the nodes are operational and active in the cluster. In this environment, any change to a value of a monitored resource will be propagated to all the other nodes in the cluster administrative domain. This processing is asynchronous to the original change, but will result in consistent values for the enrolled resources across the administrative domain. In this situation, the global status will be *consistent*, the change will be successfully made on each node, and the value of the resource on each node will match the global value for the resource.

Certain resource actions are not expected to be done when a resource is being synchronized by a cluster administrative domain. If the resource represented by an MRE is a system object, it should not be deleted, renamed or moved to a different library without removing the MRE first. If a resource is deleted, renamed or moved to a different library, the global status for the MRE will be inconsistent and any changes made to the resource on any node after that will not be propagated to the cluster administrative domain. If you need to delete, rename, or move a monitored resource, you should remove the MRE before attempting the operation.

A monitored resource that is restored on a system that is a node within the cluster administrative domain will be changed back to match the global values that are synchronized by the administrative domain. See Manage a cluster administrative domain for more information about restoring resources in this environment.

Related concepts

"Resilient environments" on page 24
Applications running in a cluster need consistent environments to operate properly. Environment resiliency allows changes to objects and attributes to be automatically propagated throughout a cluster.

"Cluster administrative domain" on page 9
A *cluster administrative domain* is used to maintain a consistent operational environment across a subset of cluster nodes to ensure that a highly available application will behave as expected as it is switched to or failed over to backup nodes.

Related tasks

"Creating a cluster administrative domain" on page 121
A cluster administrative domain can be created by using iSeries Navigator or the Create Cluster Administrative Domain (CRTADMDMN) command.

"Adding and removing monitored resource entries" on page 122
You can add a monitored resource entry to a cluster administrative domain that represents a resource that is managed by a cluster administrative domain.

"Managing a cluster administrative domain" on page 123
Once a cluster administrative domain is created and the appropriate monitored resource entries (MREs) are added, the cluster administrator should monitor the activity within the administrative domain to ensure that the monitored resources remain consistent. iSeries Navigator provides the user interface to monitor a cluster administrative domain. It provides the ability to list the MREs along with the global status for each resource. Detailed information can be displayed by selecting an MRE. This information includes the global value for each attribute associated with the MRE, along with an indication whether the attribute is consistent or inconsistent with the domain.

Attributes that can be monitored:

A monitored resource entry can be added to the cluster administrative domain for various types of resources. This topic lists the attributes that each resource type can monitor.

Resource types:
- Classes
- Independent disk pools device descriptions
- Job descriptions
- Network attributes
- System environment variables
- System values
- TCP/IP attributes
- User profiles

Table 1. Attributes that can be monitored for Classes

Attribute Name	**Description**
CPUTIME	Maximum CPU time
DFTWAIT	Default wait time
MAXTHD	Maximum threads
MAXTMPSTG	Maximum temporary storage
RUNPTY	Run priority
TIMESLICE	Time slice

Table 2. Attributes that can be monitored for Independent disk pools device descriptions

Attribute Name	**Description**
MSGQ	Message queue
RDB	Relational database
RSRCNAME	Resource name

Table 3. Attributes that can be monitored for Job descriptions

Attribute Name	**Description**
ACGCDE	Accounting code
ALWMLTTHD	Allow multiple threads
DDMCNV	DDM conversation
DEVRCYACN	Device recovery action
ENDSEV	End severity
HOLD	Hold on job queue
INLASPGRP	Initial ASP group
INQMSGRPY	Inquiry message reply
JOBMSGQFL	Job message queue full action
JOBMSGQMX	Job message queue maximum size
JOBPTY	Job priority (on JOBQ)
JOBQ	Job queue
LOG	Message logging
LOGCLPGM	Log CL program commands
OUTPTY	Output priority (on OUTQ)
OUTQ	Output queue

Table 3. Attributes that can be monitored for Job descriptions (continued)

Attribute Name	Description
PRTDEV	Print device
PRTTXT	Print text
RQSDTA	Request data or command
RTGDTA	Routing data
SPLFACN	Spooled file action
SWS	Job switches
SYNTAX	CL syntax check
TSEPOOL	Time slice end pool
USER	User

Table 4. Attributes that can be monitored for Network attributes

Attribute Name	Description
ALWADDCLU	Allow add to cluster
DDMACC	DDM/DRDA request access
NWSDOMAIN	Network server domain
PCSACC	Client request access
Note: Each attribute is treated as its own entry. For these, the resource and attribute names are identical.	

Table 5. Attributes that can be monitored for System environment variables

Any *SYS level environment variable can be monitored. The attribute and resource name are both the same as the environment variable's name.
Note: Each attribute is treated as its own entry. For these, the resource and attribute names are identical.

Table 6. Attributes that can be monitored for System values

Attribute Name	Description
QACGLVL	Accounting level
QASTLVL	Assistance level
QCCSID	Coded character set identifier
QCHRID	Default graphic character set and code page used for displaying or printing data
QCHRIDCTL	Character identifier control for the job
QCURSYM	Currency symbol
QDATFMT	Date format
QDATSEP	Date separator
QDECFMT	Decimal format
QDEVNAMING	Device naming convention
QDSPSGNINF	Controls the display of sign-on information
QJOBMSGQFL	Job message queue full action
QJOBMSGQMX	Job message queue maximum size
QJOBMSGQSZ	Initial size of job message queue in kilobytes (KB)
QJOBMSGQTL	Maximum size of job message queue (in KB)

Table 6. Attributes that can be monitored for System values (continued)

Attribute Name	Description
QJOBSPLA	Initial size of spooling control block for a job (in bytes)
QKBDBUF	Keyboard buffer
QKBDTYPE	Keyboard language character set
QLANGID	Default language identifier
QLMTSECOFR	Limit security officer device access
QMAXACTLVL	Maximum activity level of the system
QMAXJOB	Maximum number of jobs that are allowed on the system
QMAXSGNACN	The system's response when the limit imposed by QMAXSIGN system value is reached
QMAXSIGN	Maximum number of not valid sign-on attempts allowed
QPRTKEYFMT	Print key format
QPRTTXT	Up to 30 characters of text that can be printed at the bottom of listings and separator pages
QPWDEXPITV	Number of days for which a password is valid
QPWDLMTACJ	Limits the use of adjacent numbers in a password
QPWDLMTCHR	Limits the use of certain characters in a password
QPWDLMTREP	Limits the use of repeating characters in a password
QPWDLVL	Password level
QPWDMAXLEN	Maximum number of characters in a password
QPWDMINLEN	Minimum number of characters in a password
QPWDPOSDIF	Controls the position of characters in a new password
QPWDRQDDGT	Require a number in a new password
QPWDRQDDIF	Controls whether the password must be different than the previous passwords
QRETSVRSEC	Retain server security data indicator
QSECURITY	System security level
QSFWERRLOG	Software error log
QSPLFACN	Spooled file action
QTIMSEP	Time separator
QTSEPOOL	Indicates whether interactive jobs should be moved to another main storage pool when they reach time slice end
Note: Each attribute is treated as its own entry. For these, the resource and attribute names are identical.	

Table 7. Attributes that can be monitored for TCP/IP attributes

Attribute Name	Description
ARPTIMO	ARP cache timeout
ECN	Enable ECN
IPDEADGATE	IP dead gateway detection
IPDTGFWD	IP datagram forwarding
IPPATHMTU	Path MTU discovery

Table 7. Attributes that can be monitored for TCP/IP attributes (continued)

Attribute Name	**Description**
IPQOSBCH	IP QoS datagram batching
IPQOSENB	IP QoS enablement
IPQOSTMR	IP QoS timer resolution
IPRSBTIMO	IP reassembly time-out
IPSRCRTG	IP source routing
IPTTL	IP time to live (hop limit)
LOGPCLERR	Log protocol errors
NFC	Network file cache
TCPCLOTIMO	TCP time-wait timeout
TCPCNNMSG	TCP close connection message
TCPKEEPALV	TCP keep alive
TCPMINRTM	TCP minimum retransmit time
TCPR1CNT	TCP R1 retransmission count
TCPR2CNT	TCP R2 retransmission count
TCPRCVBUF	TCP receive buffer size
TCPSNDBUF	TCP send buffer size
TCPURGPTR	TCP urgent pointer
UDPCKS	UDP checksum
Note: Each attribute is treated as its own entry. For these, the resource and attribute names are identical.	

Table 8. Attributes that can be monitored for User profiles

Attribute Name	**Description**
ACGCDE	Accounting code
ASTLVL	Assistance level
ATNPGM	Attention program
CCSID	Coded character set ID
CHRIDCTL	Character identifier control
CNTRYID	Country or region ID
CURLIB	Current library
DLVRY	Delivery
DSPSGNINF	Display sign-on information
GID	Group ID number
GRPAUT	Group authority
GRPAUTTYP	Group authority type
GRPPRF	Group profile
INLMNU	Initial menu
INLPGM	Initial program to call
JOBD	Job description
KBDBUF	Keyboard buffering
LANGID	Language ID

Table 8. Attributes that can be monitored for User profiles (continued)

Attribute Name	Description
LCLPWDMGT	Local password management
LMTCPB	Limit capabilities
LMTDEVSSN	Limit device sessions
MAXSTG	Maximum allowed storage
MSGQ	Message queue
OUTQ	Output queue
OWNER	Owner
PASSWORD	User password
PRTDEV	Print device
PTYLMT	Highest schedule priority
PWDEXP	Set password to expired
PWDEXPITV	Password expiration interval
SETJOBATR	Locale job attributes
SEV	Severity code filter
SPCAUT	Special authority
SPCENV	Special environment
SRTSEQ	Sort sequence
STATUS	Status
SUPGRPPRF	Supplemental groups
UID	User ID number
USRCLS	User class
USROPT	User options

Cluster resource group exit programs

A *cluster resource group exit program* is called after a cluster-related event for a CRG occurs.

The exit program is optional for a resilient device CRG but is required for the other CRG types. When a cluster resource group exit program is used, it is called on the occurrence of cluster-wide events, including when:

- A node leaves the cluster unexpectedly.
- A node leaves the cluster as a result of the End Cluster Node (QcstEndClusterNode) API or Remove Cluster Node Entry (QcstRemoveClusterNodeEntry) API.
- The cluster is deleted as a result of the Delete Cluster (QcstDeleteCluster) API.
- A node is activated by the Start Cluster Node (QcstStartClusterNode) API.
- Communication with a partitioned node is re-established.

Exit programs are written or provided by cluster middleware IBM® Business Partners and by cluster-aware application program providers.

For detailed information on the cluster resource group exit programs, including what information is passed to them for each action code, see Cluster Resource Group Exit Program in the cluster API documentation.

Recovery domain

A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

A domain represents those nodes of the cluster from which cluster resource can be accessed. This subset of cluster nodes that is assigned to a particular cluster resource group either supports the primary point of access, secondary (backup) point of access, replicate point of access, or peer point of access.

The four types of roles a node can have in a recovery domain are:

Primary
The cluster node that is the primary point of access for the resilient cluster resource.

- For a data CRG, the primary node contains the principle copy of a resource.
- For an application CRG, the primary node is the system on which the application is currently running.
- For a device CRG, the primary node is the current owner of the device resource.

  Note: When using geographic mirroring, the nodes in the recovery domain of a device CRG require a site name and data port IP addresses. See Site name and data port IP addresses for details.
- For a peer CRG, the primary node is not supported.

If the primary node for a CRG fails, or a manual switchover is initiated, then the primary point of access for that CRG is moved to the first backup node

Backup
The cluster node that will take over the role of primary access if the present primary node fails or a manual switchover is initiated.

- For a data CRG, this cluster node contains a copy of that resource which is kept current with replication.
- For a peer CRG, the backup node is not supported.

Replicate
A cluster node that has copies of cluster resources, but is unable to assume the role of primary or backup. Failover or switchover to a replicate node is not allowed. If you ever want a replicate node to become a primary, you must first change the role of the replicate node to that of a backup node.

- For peer CRGs, nodes defined as replicate represent the inactive access point for cluster resources.

Peer A cluster node which is not ordered and can be an active access point for cluster resources. When the CRG is started, all the nodes defined as peer will be an active access point.

- For a peer CRG, the access point is controlled entirely by the management application and not the system. The peer role is only supported by the peer CRG.

Primary-backup model

For nodes that participate in a primary-backup model, each node in the recovery domain has a role with respect to the current operational environment of the cluster. This is called its *current role* in the recovery domain. As the cluster goes through operational changes such as nodes ending, nodes starting, and nodes failing, the node's current role is changed accordingly. Each node in the recovery domain also has a role with respect to the preferred or ideal cluster environment. This is called its *preferred role* in the recovery domain. The preferred role is a static definition that is initially set when the cluster resource group is created. As the cluster environment changes, this role is not changed. The preferred role is only changed when nodes are added or removed from the recovery domain, or when a node is removed from the cluster. You can also manually change the preferred roles.

Conceptually, you can view the recovery domain within a primary-backup model as follows:

Table 9. Node roles for primary-backup CRGs

Node	**Current role**	**Preferred role**
A	Backup 1	Primary
B	Backup 2	Backup 1
C	Primary	Backup 2
D	Replicate	Replicate

In this example, Nodes A, B, C, and D provide an example of a CRG that is a primary-backup model. Node C is serving as the current primary node. Because it has a preferred role of the second backup, Node C's current role as primary results from two failover or switchover actions. Upon the first failover or switchover action, the primary role moved from Node A to Node B since Node B is defined as the first backup. The second failover or switchover triggered Node C to become the primary node since it is defined as the second backup node. Node D current and preferred role is that of replicate. A replicate node cannot be the assume the point of access during a failover or switchover unless its role is changed manually to either primary or backup.

Note: The role of each node in the recovery domain can also be changed manually. The example illustrates how the roles in the recovery domain change when switchover or failover actions occur and no manual changes are made to the designation of the roles in the recovery domain.

Peer model

For peer model, a node within a cluster resource groups can have one of two roles: peer or replicate.

Table 10. Node roles for peer CRGs

Node	**Current role**	**Preferred role**
A	Peer	Peer
B	Peer	Peer
C	Peer	Peer
D	Replicate	Replicate

Nodes A, B, and C are defined in the recovery domain as peer nodes. When a failure occurs on Node A, it is communicated to all nodes in recovery domain regardless of current role. These nodes resumes the operation at the point when Node A failed. Node D contains the data, but cannot resume the operation since it is defined as Replicate.

Any number of nodes can be designated as the peer or replicate. Peer nodes are not ordered and can become an active access point for the cluster resources. Replicates are not ordered and cannot become an active access point for the cluster resource unless the Change Cluster Resource Group (QcstChangeClusterResourceGroup) API is used to change its role from replicate to peer.

Related tasks

"Changing the recovery domain for a cluster resource group" on page 118
You can change the roles of nodes in a recovery domain for a cluster resource group, as well as add or remove nodes from a recovery domain. For a device cluster resource group, you can also change site name and data port IP addresses for a node in the recovery domain.

"Performing a switchover" on page 118
Performing a manual switchover causes the current primary node to switch over to the backup node, as defined in the cluster resource group's recovery domain.

Cluster version

A *cluster version* represents the level of function available on the cluster.

Versioning is a technique that allows the cluster to contain nodes at multiple release levels and fully interoperate by determining the communications protocol level to be used. If you are using a cluster that will contain nodes of varying release levels, see Multiple-release clusters.

There are actually two cluster versions:

Potential cluster version
: Represents the most advanced level of cluster function available for a given node. This is the version at which the node is capable of communicating with the other cluster nodes.

Current cluster version
: Represents the version currently being used for all cluster operations. This is the version of communications between the nodes in the cluster.

The potential cluster version is incremented on every i5/OS release which has significant new clustering functionality not available in earlier cluster versions. If the current cluster version is less than the potential cluster version, then that function cannot be used since some nodes cannot recognize or process the request. To take advantage of such new function, every node in the cluster needs to be at the same potential cluster version and the current cluster version must also be set to that level.

When a node attempts to join a cluster, its potential cluster version is compared against the current cluster version. If the value of the potential cluster version is not the same as current (N) or not equal to the next version level (N+1), then the node is not allowed to join the cluster. Note that the current cluster version is initially set by the first node defined in the cluster using the value specified on the create cluster API or command.

For example if you want V5R3 nodes to exist with V5R4 nodes you can do one of the following:

- Create the cluster on a V5R3 node and add in the V5R4 node.
- Create the cluster on a V5R4 node specifying to allow previous nodes to be added to the cluster, then add V5R3 nodes to the cluster.

In a multiple-release cluster, cluster protocols will always be run at the lowest node release level, the current cluster version. This is defined when the cluster is first created. N can either be set to the potential node version running on the node that originated the create cluster request or one cluster version previous to the originators potential node version. Nodes in the cluster can differ by at most one cluster version level.

Once all nodes in the cluster have been upgraded to the next release, the cluster version can be upgraded so that new functions are available. This can be accomplished by adjusting the cluster version.

Attention: If the new version of the operating system is not equal or one version higher to the current cluster version, then the cluster node will fail when it is restarted. To recover from this situation, the cluster on that node must be deleted and the cluster version adjusted before the node can be re-added to the cluster.

Attention: When you are using switchable independent disk pools in your cluster, there are restrictions in performing switchover between releases. You need to switch a previous release independent disk pool to a node running the current release of i5/OS and make it available. After it is made available on the node running the current release of i5/OS, its internal contents are changed and it cannot be made available to the previous release node again.

Read more on cluster versions in the Cluster APIs documentation, including information about restrictions and how cluster versions correspond to i5/OS releases.

Upgrading to a new release

If you are upgrading a node to new release, you must ensure that the node you are upgrading has the appropriate cluster version. Clustering only supports nodes with one version difference in the same cluster. If you are upgrading in a cluster environment where all nodes are at the same release, you should upgrade all the nodes to the new release before changing the cluster version. This ensures that all functions associated to the new release will be available to you.

Before performing the actions associated with the upgrade, you should first determine the current cluster version for your cluster. You can display pertinent information about your cluster, including the cluster version, by using the Display Cluster Information (DSPCLUINF) command. You can also determine your current cluster version by using iSeries Navigator. See the topic, Adjust cluster version for instructions on verifying your current cluster version or changing the cluster version with iSeries Navigator.

The following table provides actions you will need to take when performing an upgrade in a cluster environment.

Table 11. Upgrading nodes to V5R4

Current release of node you are upgrading	Current cluster version	Actions
V5R3	4	1. Upgrade the node to V5R4. 2. Start the upgraded node.
V5R3	3	1. Change the cluster version to 4. 2. Upgrade the node to V5R4. 3. Start the upgraded node. **Note:** If other nodes in your cluster are at V5R2, see Scenarios 4 and 5 below for details.
V5R2	less than or equal to 3	**Option A** 1. Remove the node that is being upgraded from the cluster. 2. Change the cluster version to 4. 3. Upgrade the node to V5R4. 4. Add node back into the cluster. **Option B** 1. Upgrade all nodes to V5R3. 2. Change the cluster version to 4. 3. Upgrade all nodes to V5R4.

Scenarios

The following scenarios can help you determine what you need to do to perform an upgrade in your cluster environment. Before performing the upgrade or any actions you should first determine the current cluster version for your cluster.

Scenario 1: Node to be upgraded is at V5R3. At least one other node in the cluster is at V5R3. Current cluster version is 4.

Action: Upgrade node to V5R4. After upgrading the node, start clustering on the upgraded node.

Scenario 2: Node to be upgraded is at V5R3. At least one other node in the cluster is at V5R3. Current cluster version is 3.
Action: Change current cluster version to 4. Upgrade the node to V5R4. Start clustering on the upgraded node.

Scenario 3: Node to be upgraded is V5R2. At least one other node in the cluster is at V5R3. Current cluster version is 3.
Action: Remove the node being upgraded to V5R4 from the cluster before the upgrade. Change the current cluster version to 4. Upgrade the node to V5R4 and add it back into the cluster.

Scenario 4: Node to be upgraded is at V5R3. Currently have V5R2 and V5R3 nodes in cluster. Current cluster version is 3. V5R3 node being upgraded to V5R4 is LESS important than the nodes staying at V5R2.
Actions:

1. Remove the node that is being upgraded from the cluster.
2. Upgrade node to V5R4.
3. Upgrade the remaining V5R2 nodes to at least V5R3.
4. Change the cluster version to 4.
5. Add the upgraded node back into the cluster.

Scenario 5: Node to be upgraded is at V5R3. Currently have V5R2 and V5R3 nodes in cluster. Current cluster version is 3. V5R3 node being upgraded to V5R4 is MORE important than the nodes staying at V5R2.
Actions:

1. Remove all the V5R2 nodes from the cluster.
2. Change the cluster version to 4.
3. Upgrade the node to V5R4.
4. Start the upgraded node.
5. As the remaining V5R2 nodes are upgraded to V5R3, they can be added back into the cluster.

Scenario 6: Node to be upgraded is at V5R2. At least one other node in the cluster is at V5R2. Current cluster version is less than or equal to 3.
Action: Upgrade all the nodes to V5R3. Change the cluster version to 4. Upgrade all nodes to V5R4.

Related concepts

"Multiple-release clusters" on page 96
If creating a cluster that will include nodes at multiple cluster versions, then certain steps are required when you create the cluster.

"Common cluster problems" on page 152
Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

Related tasks

"Creating a cluster" on page 111
To create and configure a cluster, you need to include at least one node in the cluster and you must have access to at least one of the nodes that will be in the cluster.

"Adjusting the cluster version of a cluster" on page 115
The cluster version defines the level at which all the nodes in the cluster are actively communicating with each other.

Resilient resources

Resilient resources are system resources, such as data, devices and applications, that are highly available if you have used clustering on your systems.

If a cluster node that is the primary access point for a particular set of resilient resources in the cluster should incur an outage, another cluster node that is defined as the backup for that set of resources now becomes the access point.

The types of system resources that can be resilient are:

1. Data being replicated between nodes.
2. Applications using an IP address, which can be switched from one node to another.
3. Hardware devices which can be switched from one node to another.
4. Peer resources that are supported by a cluster administrative domain. |

The definition of the relationship between the nodes that are associated with a set of resilient resources is found in the *cluster resource group (CRG)* object. Cluster resource groups are replicated and coordinated across the nodes in the cluster through cluster resource services.

Related concepts

"Cluster resource group" on page 8
A *cluster resource group (CRG)* is an i5/OS system object that is a set or grouping of cluster resources that are used to manage events that occur in a clustered environment. The cluster resource group describes a recovery domain and supplies the name of the cluster resource group exit program which gets called when certain cluster events occur.

Resilient applications:

A *resilient application* is an application that can be restarted on a different cluster node without requiring you to reconfigure the clients.

See Making application programs resilient to learn about what characteristics make an application resilient.

A resilient application needs the ability to recognize the temporary loss of the Internet Protocol (IP) connection between the client and the server. The client application must be aware that the IP connection will be temporarily unavailable and must retry access rather than ending or initiating a failover. Similarly, if you are performing a switchover, server applications need to be aware that the IP connection is no longer available. Eventually, an error condition is returned to the server application. Once this error condition is received, it is best if the server application recognizes the condition and ends normally.

IP address takeover is a high availability function that is used to protect clients from application server outages. An **application takeover IP address** is a floating address that is to be associated with an application. The concept is to use IP address aliasing to define a floating IP address that is associated with multiple application servers or hosts. When one application server in a cluster fails, another cluster node assumes the responsibilities of the application server without requiring you to reconfigure the clients.

Also introduced in support of IP address takeover is the concept of application cluster resource groups (CRGs). Application CRGs are cluster resource groups that contain an application takeover IP address resource and a recovery domain. The recovery domain contains the list of application servers within the cluster that support a particular application. If a single resource fails, cluster resource services initiates a failover on the group to which the failing resource belongs.

Related concepts

"Making application programs resilient" on page 39
Learn how to make application programs resilient.

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

Related tasks

"Cluster applications" on page 38
Application resilience is one of the key elements in a clustered environment. If you are planning to write and use highly available applications in your cluster you should be aware that these applications have specific availability specifications.

Resilient data:

Resilient data is data that is replicated (copied) on more than one node in a cluster.

Each node in the recovery domain contains a copy of the resilient data maintained through some replication mechanism. Nodes that are defined as backup in the recovery domain can assume the role of primary point of access for the resilient data. Nodes that are defined as replicates also contain a copy of the data, but cannot assume the role of primary. Typically, data copied to a replicate node is used to offload work, such as backup or read-only queries, from the primary node.

Related concepts

"Replication" on page 34
Replication makes a copy of something in real time. It means copying objects from one node in a cluster to one or more other nodes in the cluster.

Resilient devices:

Resilient devices are physical resources, represented by a configuration object, such as a device description, that are accessible from more than one node in a cluster.

| In the event of an outage, the point of access for the resource is switched to the first backup node in the
| cluster resource group recovery domain. Independent disk pools, or independent ASPs are resilient
| devices that can go offline or come online without impacting the rest of system storage. Additionally you
| can use geographic mirroring which is a subfunction of cross-site mirroring (XSM) which is part of i5/OS
| Option 41, High Available Switchable Resources. Geographic mirroring is a function that keeps two
| identical copies of an independent disk pool at two sites to provide high availability and disaster
| recovery. The copy owned by the primary node is the production copy and the copy owned by a backup
| node at the other site is the mirror copy. User operations and applications access the independent disk
| pool on the primary node, the node that owns the production copy.

A *resilient device cluster resource group* can contain a list of switchable devices. Each device in the list identifies a switchable independent disk pool. The entire collection of devices are switched to the backup node when an outage occurs. Optionally, the devices can also be varied on as part of the switchover/failover process. There are limitations related to the physical configuration associated with the list of switchable devices. See Independent disk pools for more information about how to set up the appropriate configuration for an independent disk pool defined to be resilient.

A resilient device CRG is very similar to the other types of CRGs. One difference, the list of switchable devices, was mentioned above. Another difference is that the exit program is optional for a device CRG. If environment or data specific processing is needed, an exit program can be used for the CRG. See the Create Cluster Resource Group (QcstCreateClusterResourceGroup) API for additional information about this type of CRG.

Resilient environments:

Applications running in a cluster need consistent environments to operate properly. Environment resiliency allows changes to objects and attributes to be automatically propagated throughout a cluster.

| The consistency and propagation of attributes and objects within an environment is controlled and
| maintained through the cluster administrative domain. The cluster administrative domain is based on a
| peer CRG and enables synchronization of changes to system objects and configuration parameters across

| nodes. These synchronized objects or parameters are called monitored resources. They are represented by
| monitored resource entries in the cluster administrative domain.

Related concepts

"Cluster administrative domain" on page 9
A *cluster administrative domain* is used to maintain a consistent operational environment across a subset of cluster nodes to ensure that a highly available application will behave as expected as it is switched to or failed over to backup nodes.

"Monitored resources" on page 10
Monitored resources are types of system resources that can be managed by a cluster administrative domain. These resources are represented in the cluster administrative domain as *monitored resource entries (MREs)*.

Related tasks

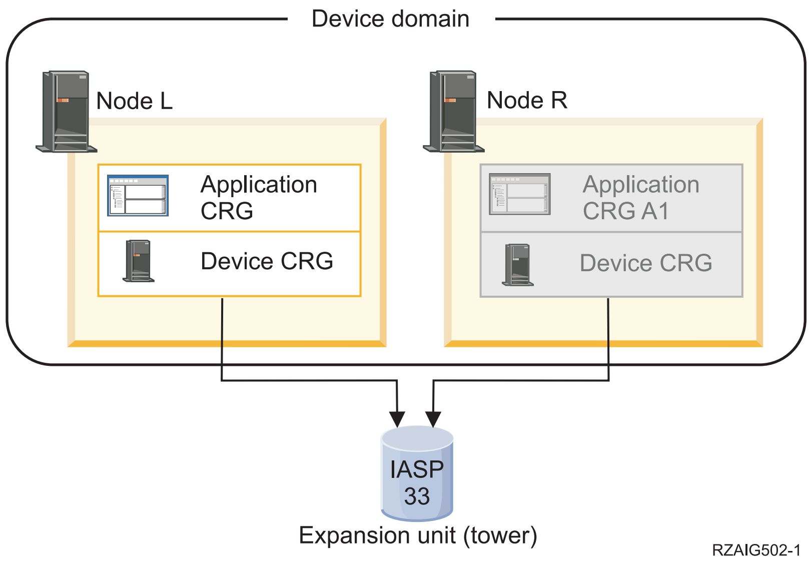
| "Creating a cluster administrative domain" on page 121
A cluster administrative domain can be created by using iSeries Navigator or the Create Cluster
| Administrative Domain (CRTADMDMN) command.

Device domains

A *device domain* is a subset of nodes in a cluster that share device resources. More specifically, nodes in a device domain can participate in a switching action for some collection of resilient device resources.

Device domains are identified and managed through a set of interfaces that allow you to add a node to a device domain or remove a node from a device domain.

Device domains are used to manage certain global information necessary to switch a resilient device from one node to another. All nodes in the device domain need this information to ensure that no conflicts occur when devices are switched. For example, for a collection of switchable independent disk pools, the independent disk pool identification, disk unit assignments, and virtual address assignments must be unique across the entire device domain.



A cluster node can belong to at most one device domain. Before a node can be added to the recovery domain for a device CRG, the node must be first defined as a member of a device domain. All nodes that will be in the recovery domain for a device CRG must be in the same device domain.

To create and manage device domains, you must have Option 41 (i5/OS - HA Switchable Resources) installed and a valid license key on your system.

Related concepts

"Example: A switched disk cluster using independent disk pools" on page 136
A cluster using switched disk technology provides an alternative to having the data replicated. In a switched disk cluster, the data is actually contained in independent disk pools (also referred to as independent ASPs).

Related tasks

"Adding a node to a device domain" on page 119
A device domain is a subset of nodes in a cluster that share device resources.

"Removing a node from a device domain" on page 120
A device domain is a subset of nodes in a cluster that share device resources.

Option 41 (HA Switchable Resources):

To create and manage device domains, you must have Option 41 (i5/OS - HA Switchable Resources) installed and a valid license key on your system.

You need to have this feature installed if you want to do any of the following in your clustered environment:

- Use the iSeries Navigator cluster management interface.
- Switch independent disk pools (independent ASPs) between systems.
- Cross-site mirroring between geographically dispersed systems

Related tasks

"Adding a node to a device domain" on page 119
A device domain is a subset of nodes in a cluster that share device resources.

"Removing a node from a device domain" on page 120
A device domain is a subset of nodes in a cluster that share device resources.

Cluster events

Within a cluster several types events, actions, and services can occur.

Failover

Failover occurs when a server in a cluster automatically switches over to one or more backup servers in the event of a system failure.

Contrast this with a switchover, which happens when you manually switch access from one server to another. A switchover and a failover function identically once they have been triggered. The only difference is how the event is triggered.

When a failover occurs, access is switched from the cluster node currently acting as the primary node in the recovery domain of the cluster resource group to the cluster node designated as the first backup. See recovery domain for information about how the switchover order is determined.

When multiple cluster resource groups (CRGs) are involved in a failover action, the system processes the device CRGs (switchable devices) first, the data CRGs (switchable data groups) second, and the application CRGs (switchable applications) last.

The failover message queue receives messages regarding failover activity. You can use it to control the failover processing of a cluster resource group.

Related concepts

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

"Cluster resource group" on page 8
A *cluster resource group (CRG)* is an i5/OS system object that is a set or grouping of cluster resources

that are used to manage events that occur in a clustered environment. The cluster resource group describes a recovery domain and supplies the name of the cluster resource group exit program which gets called when certain cluster events occur.

"Failover message queue" on page 132
The failover message queue receives messages regarding failover activity.

"Hardware requirements for clusters" on page 91
Any System i model that is capable of running OS/400® V4R4, or later, is compatible for using clustering.

Related tasks

"Switchover" on page 29
Switchover happens when you manually switch access to a resource from one server to another.

Example: Failure:

Usually, a failover results from a node failure, but there are other reasons that can also generate a failover.

It is possible for a problem to affect only a single cluster resource group that can cause a failover for that cluster resource group (CRG) but not for any other CRG.

The following table shows various types of failures and the category they fit in:

Failure	General category
CEC hardware failure (CPU, for example)	2
Communications adapter, line, or router failure; or ENDTCPIFC affecting all IP interface addresses defined for the node	4
Power loss to the CEC	1
Operating system software machine check	2
ENDTCP(*IMMED or *CNTRLD with a time limit) is issued	1
ENDSBS QSYSWRK(*IMMED or *CNTRLD) is issued	1
ENDSBS(*ALL, *IMMED, or *CNTRLD) is issued	1
ENDSYS (*IMMED or *CNTRLD) is issued	1
PWRDWNSYS(*IMMED or *CNTRLD) is issued	1
Initial program load (IPL) button is pushed while cluster resource services is active on system	1
Cancel Job (*IMMED or *CNTRLD with a time limit) of the QCSTCTL job is issued	1
Cancel Job (*IMMED or *CNTRLD with a time limit) of the QCSTCRGM job is issued	1
Cancel Job (*IMMED or *CNTRLD with a time limit) of a cluster resource group job is issued	3
End Cluster Node API is called	1
Remove Cluster Node API is called	1
Cluster resource group job has a software error that causes it to end abnormally	3
Enters function 8 or function 3 from control panel to power down system	2

Failure	General category
Enters function 7 for a delayed power down of a partition	1
Application program failure for an application cluster resource group	3

General category:

1. All of cluster resource services (CRS) fails on a node and is detected as a node failure. The node may actually be operational or the node may have failed (for example, a system failure due to power loss). If all of cluster resource services fails, then the resources that are managed by CRS will go through the failover process.
2. All of CRS fails on a node but it is detected as a cluster partition. The node may or may not be operational.
3. A failure occurs on an individual cluster resource group. These conditions are always detected as a failure.
4. A failure occurs but the node and cluster resource services are still operational and it is detected as a cluster partition.

When a failure occurs, the action taken by cluster resource services for a specific cluster resource group depends on the type of failure and the state of the cluster resource group. However, in all cases, the exit program is called. A failover may must work with a list of failed nodes. When the exit program is called, it needs to determine if it must deal with only a single node failure or with a list of failed nodes.

If the cluster resource group is *inactive*, the membership status of the failed node in the cluster resource group's recovery domain is changed to either an *Inactive* or *Partition* status. However, the node roles are not changed, and the backup nodes are not reordered. The backup nodes are reordered in an inactive cluster resource group when the Start Cluster Resource Group (STRCRG) command or the Start Cluster Resource Group (QcstStartClusterResourceGroup) API is called. But, the Start Cluster Resource Group API will fail if the primary node is not active. You must issue the Change Cluster Resource Group (CHGCRG) command or the Change Cluster Resource Group (QcstChangeClusterResourceGroup) API to designate an active node as the primary node, then call the Start Cluster Resource Group API again.

If the cluster resource group is *active* and the failing node is *not* the primary node, the failover updates the status of the failed recovery domain member in the cluster resource group's recovery domain. If the failing node is a backup node, the list of backup nodes is reordered so that active nodes are at the beginning of the list.

If the cluster resource group is *active* and the recovery domain member is the primary node, the following actions are performed based on which type of failure has occurred:

Category 1 Failure
: Failover occurs. The primary node is marked *inactive* in each cluster resource group and made the last backup node. The node that was the first backup becomes the new primary node. All device cluster resource groups failover first. Then, all data cluster resource groups failover. Finally, all application cluster resource groups failover. If a failover for any CRG detects that none of the backup nodes are active, the status of the CRG is set to *indoubt*.

Category 2 Failure
: Failover occurs but the primary node does not change. All nodes in the cluster partition that do not have the primary node as a member of the partition will end the active cluster resource group. The status of the nodes in the recovery domain in the cluster resource group are set to a *partition* status for each node that is in the primary partition. If a node really failed but is detected only as a partition problem and the failed node was the primary node, you lose all the data and application services on that node and no automatic failover is started. You must either declare the node as failed or bring the node back up and start clustering on that node again. See Change partitioned nodes to failed for more information.

Category 3 Failure
: If only a single cluster resource group is affected, failover occurs on an individual basis because

cluster resource groups are independent of each other. It may happen that several cluster resource groups are affected at the same time due to someone canceling several cluster resource jobs. However, the type of failure is handled on a CRG by CRG basis, and no coordinated failover between CRGs is performed. The primary node is marked as *inactive* in each cluster resource group and made the last backup node. The node that was the first backup node becomes the new primary node. If there is no active backup node, the status of the cluster resource group is set to *indoubt*.

Category 4 Failure

This category is similar to category 2. However, while all nodes and cluster resource services on the nodes are still operational, not all nodes can communicate with each other. The cluster is partitioned, but the primary node or nodes are still providing services. However, because of the partition, you may experience various problems. For example, if the primary node is in one partition and all the backup nodes or replicate nodes are in another partition, you are no longer replicating data and have no protection should the primary node fail. In the partition that contains the primary node, the failover process updates the status of the nodes in the cluster resource group's recovery domain to *partition* for all nodes in the other partition. In the partition that does not contain the primary node, the status of the nodes in the cluster resource group's recovery domain for all nodes in the other partition is set to *partition.*

Related concepts

"Partition errors" on page 154
Certain cluster conditions are easily corrected. If a cluster partition has occurred, you can learn how to recover. This topic also tells you how to avoid a cluster partition and gives you an example of how to merge partitions back together.

Switchover

Switchover happens when you manually switch access to a resource from one server to another.

You usually initiate a manual switchover if you wanted to perform system maintenance, such as applying program temporary fixes (PTFs), installing a new release, or upgrading your system. Contrast this with a failover, which happens automatically when a outage occurs on the primary node.

When a switchover occurs, access is switched from the cluster node currently acting as the primary node in the recovery domain of the cluster resource group to the cluster node designated as the first backup. See recovery domain for information on how the switchover order is determined.

If you are doing an administrative switchover of multiple CRGs, the order you specify should consider the relationships between the CRGs. For example, if you have an application CRG that depends on data associated with a device CRG, the steps to an ordered switchover are:

1. Stop the application on the old primary node (to quiesce changes to the data).
2. Switch the device CRG to the new primary node.
3. Switch the application CRG to the new primary node.
4. Restart the application on the new primary node.

Related concepts

"Failover" on page 26
Failover occurs when a server in a cluster automatically switches over to one or more backup servers in the event of a system failure.

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

Related tasks

"Performing a switchover" on page 118
Performing a manual switchover causes the current primary node to switch over to the backup node, as defined in the cluster resource group's recovery domain.

Rejoin

Rejoin means to become an active member of a cluster after having been a nonparticipating member.

For example, when clustering is restarted on a node after the node has been inactive, the cluster node rejoins the cluster. You start cluster resource services on a node by starting it from a node that is already active in the cluster. Beginning with cluster version 3, a node can start itself and will be able to rejoin the current active cluster, provided it can find an active node in the cluster. See Start a cluster node for details.

Suppose that nodes A, B, and C make up a cluster. Node A fails. The active cluster is now nodes B and C. Once the failed node is operational again, it can rejoin the cluster when the node is started from any cluster node, including itself. The rejoin operation is done on a cluster resource group basis, which means that each cluster resource group (CRG) joins the cluster independently.

The primary function of rejoin ensures that the CRG object is replicated on all active recovery domain nodes. The rejoining node, as well as all existing active cluster nodes, must have an identical copy of the CRG object. In addition, they must have an identical copy of some internal data.

When a node fails, the continued calling of cluster resource services on the remaining nodes in the cluster can change the data in a CRG object. The modification must occur due to the calling of an API or a subsequent node failure. For simple clusters, the rejoining node is updated with a copy of the CRG from some node that is currently active in the cluster. However, this may not be true in all cases.

See Starting or ending a cluster administrative domain node for information about the behavior of a cluster administrative domain during a rejoin.

Related concepts

"Starting or ending a cluster administrative domain node" on page 125
Consider the following information when starting and ending a cluster node.

Related tasks

"Starting a cluster node" on page 113
Starting a cluster node starts cluster resource services on a node in the cluster. Beginning with cluster version 3, a node can start itself and will be able to rejoin the current active cluster, provided it can find an active node in the cluster.

"Managing a cluster administrative domain" on page 123
Once a cluster administrative domain is created and the appropriate monitored resource entries (MREs) are added, the cluster administrator should monitor the activity within the administrative domain to ensure that the monitored resources remain consistent. iSeries Navigator provides the user interface to monitor a cluster administrative domain. It provides the ability to list the MREs along with the global status for each resource. Detailed information can be displayed by selecting an MRE. This information includes the global value for each attribute associated with the MRE, along with an indication whether the attribute is consistent or inconsistent with the domain.

"Changing partitioned nodes to failed" on page 156
Sometimes, a partitioned condition is reported when there really was a node outage. This can occur when cluster resource services loses communications with one or more nodes, but cannot detect if the nodes are still operational. When this condition occurs, a simple mechanism exists for you to indicate that the node has failed.

Example: Rejoin:

This topic describes the actions that can occur when a node rejoins a cluster.

The following diagram describes the actions taken whenever a node rejoins the cluster. In addition, the state of the rejoining nodes will be changed from *inactive* to *active* in the membership status field in the recovery domain of the CRG. The exit program is called on all nodes in the CRG's recovery domain and is passed an action code of Rejoin.

Table 12. Rejoin operation

Rejoin operation			
Rejoining node		Cluster nodes	
Contains copy of CRG	Does not contain copy of CRG	Contain copy of CRG	Do not contain copy of CRG
(1)	(2)	(3)	(4)

Using the diagram above, the following situations are possible:

1. 1 and 3
2. 1 and 4
3. 2 and 3
4. 2 and 4

If a node in the cluster has a copy of the CRG, the general rule for rejoin is that the CRG is copied from an active node in the cluster to the rejoining node.

Rejoin Situation 1
A copy of the CRG object from a node in the cluster is sent to the joining node. The result is:
- The CRG object is updated on the joining node with the data sent from the cluster.
- The CRG object may be deleted from the joining node. This can occur if the joining node was removed from the CRG's recovery domain while the joining node was out of the cluster.

Rejoin Situation 2
A copy of the CRG object from the joining node is sent to all cluster nodes. The result is:
- No change if none of the cluster nodes are in the CRG's recovery domain.
- The CRG object may be created on one or more of the cluster nodes. This can occur in the following scenario:
  - Nodes A, B, C, and D make up a cluster.
  - All four nodes are in the recovery domain of the CRG.
  - While node A is out of the cluster, the CRG is modified to remove B from the recovery domain.
  - Nodes C and D fail.
  - The cluster is only node B which does not have a copy of the CRG.
  - Node A rejoins the cluster.
  - Node A has the CRG (although it is down level by now) and Node B does not. The CRG is created on node B. When nodes C and D rejoin the cluster, the copy of the CRG in the cluster updates node C and D and the previous change to remove node B from the recovery domain is lost.

Rejoin Situation 3
A copy of the CRG object from a node in the cluster is sent to the joining node. The result is:
- No change if the joining node is not in the CRG's recovery domain.
- The CRG object may be created on the joining node. This can occur if the CRG was deleted on the joining node while cluster resource services is not active on the node.

Rejoin Situation 4
Some internal information from one of the nodes in the cluster may be used to update information about the joining node but nothing occurs that is visible to you.

Merge

A *merge* operation is similar to a rejoin operation except that it occurs when nodes that are partitioned begin communicating again.

The partition may be a true partition in that cluster resource services is still active on all nodes. However, some nodes cannot communicate with other nodes due to a communication line failure. Or, the problem may be that a node actually failed, but was not detected as a failure.

In the first case, the partitions are merged back together automatically once the communication problem is fixed. This happens when both partitions periodically try to communicate with the partitioned nodes and eventually reestablish contact with each other. In the second case, cluster resource services must be restarted on the failed node by starting the node from any node in the cluster.

Related concepts

"Rejoin" on page 30
Rejoin means to become an active member of a cluster after having been a nonparticipating member.

"Partition errors" on page 154
Certain cluster conditions are easily corrected. If a cluster partition has occurred, you can learn how to recover. This topic also tells you how to avoid a cluster partition and gives you an example of how to merge partitions back together.

Related tasks

"Starting a cluster node" on page 113
Starting a cluster node starts cluster resource services on a node in the cluster. Beginning with cluster version 3, a node can start itself and will be able to rejoin the current active cluster, provided it can find an active node in the cluster.

"Changing partitioned nodes to failed" on page 156
Sometimes, a partitioned condition is reported when there really was a node outage. This can occur when cluster resource services loses communications with one or more nodes, but cannot detect if the nodes are still operational. When this condition occurs, a simple mechanism exists for you to indicate that the node has failed.

Example: Merge:

Merge operations can occur in several different situations.

A merge operation can occur with one of the following configurations:

Table 13. Merge between a primary and secondary partition

merge	
primary partition	secondary partition

Table 14. Merge between a secondary and secondary partition

merge	
secondary partition	secondary partition

Primary and secondary partitions are unique to cluster resource groups (CRG). For a primary-backup CRG, a primary partition is defined as the partition that contains the node identified as the primary access point. A secondary partition is defined as a partition that does not contain the node identified as the primary access point.

For peer CRG, if the recovery domain nodes are fully contained within one partition, it will be the primary partition. If the recovery domain nodes span a partition, there will be no primary partition. Both partitions will be secondary partitions.

See Starting and ending a cluster node for information about the behavior of a cluster administrative domain during a rejoin.

Table 15. Merge between a primary and secondary partition

merge operation			
primary partition		secondary partition	
contains copy of CRG	does NOT contain copy of CRG	contains copy of CRG	does NOT contain copy of CRG
(1)	(2)	(3)	(4)

During a primary secondary merge as shown in the diagram above, the following situations are possible:

1. 1 and 3
2. 1 and 4
3. 2 and 3 (Cannot happen since a majority partition has the primary node active and must have a copy of the CRG.)
4. 2 and 4 (Cannot happen since a majority partition has the primary node active and must have a copy of the CRG.)

Primary-secondary merge situations

A copy of the CRG object is sent to all nodes in the secondary partition. The following actions can result on the nodes in the secondary partition:

- No action since the secondary node is not in the CRG's recovery domain.
- A secondary node's copy of the CRG is updated with the data from the primary partition.
- The CRG object is deleted from a secondary node since the secondary node is no longer in the CRG's recovery domain.
- The CRG object is created on the secondary node since the object does not exist. However, the node is in the recovery domain of the CRG copy that is sent from the primary partition.

Table 16. Secondary and secondary merge scenario

merge operation			
secondary partition		secondary partition	
contains copy of CRG	does NOT contain copy of CRG	contains copy of CRG	does NOT contain copy of CRG
(1)	(2)	(3)	(4)

During a secondary-secondary merge as shown in the diagram above, the following situations are possible:

1. 1 and 3
2. 1 and 4
3. 2 and 3
4. 2 and 4

Secondary-secondary merge situation 1

For a primary-backup CRG, the node with the most recent change to the CRG is selected to send a copy of the CRG object to all nodes in the other partition. If multiple nodes are selected because they all appear to have the most recent change, the recovery domain order is used to select the node.

When merging two secondary partitions for peer CRG, the version of the peer CRG with Active status will copied to other nodes in the other partition. If both partitions have the same status for peer CRG, the partition which contains the first node listed in the cluster resource group recovery domain will be copied to nodes in another partition.

The following actions can occur on the receiving partition nodes in either a primary-backup CRG or a peer CRG:

- No action since the node is not the CRG's recovery domain.
- The CRG is created on the node since the node is in the recovery domain of the copy of the CRG object it receives.
- The CRG is deleted from the node since the node is not in the recovery domain of the copy of the CRG object it receives.

Secondary-secondary merge situations 2 and 3

A node from the partition which has a copy of the CRG object is selected to send the object data to all nodes in the other partition. The CRG object may be created on nodes in the receiving partition if the node is in the CRG's recovery domain.

Secondary-secondary merge situation 4

Internal data is exchanged to ensure consistency throughout the cluster.

A primary partition can subsequently be partitioned into a primary and secondary partition. If the primary node fails, Cluster Resource Service (CRS) detects it as a node failure. The primary partition becomes a secondary partition. The same result occurs if you ended the primary node that uses the End Cluster Node API. A secondary partition can become a primary partition if the primary node becomes active in the partition either through a rejoin or merge operation.

For a merge operation, the exit program is called on all nodes in the CRG's recovery domain regardless of which partition the node is in. The same action code as rejoin is used. No roles are changed as a result of the merge, but the membership status of the nodes in the CRG's recovery domain is changed from *partition* to *active*. Once all partitions merge together, the partition condition is cleared, and all CRG API's can be used.

Related concepts

"Starting or ending a cluster administrative domain node" on page 125
Consider the following information when starting and ending a cluster node.

Replication

Replication makes a copy of something in real time. It means copying objects from one node in a cluster to one or more other nodes in the cluster.

Replication makes and keeps the objects on your systems identical. If you make a change to an object on one node in a cluster, this change is replicated to other nodes in the cluster.

Related concepts

"Resilient data" on page 24
Resilient data is data that is replicated (copied) on more than one node in a cluster.

"Planning for logical replication" on page 102
Multiple copies of the data are maintained with logical replication. Data is replicated, or copied, from the primary node in the cluster to the backup nodes designated in the recovery domain. When an outage occurs on the primary node, the data remains available as a designated backup node takes over as the primary point of access.

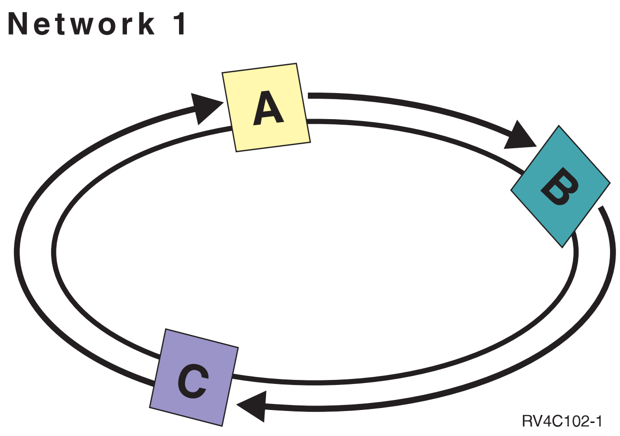
Heartbeat monitoring

Heartbeat monitoring is a cluster resource services function that ensures that each node is active by sending a signal from every node in the cluster to every other node in the cluster to convey that they are still active.

When the heartbeat for a node fails, cluster resource services takes the appropriate action.

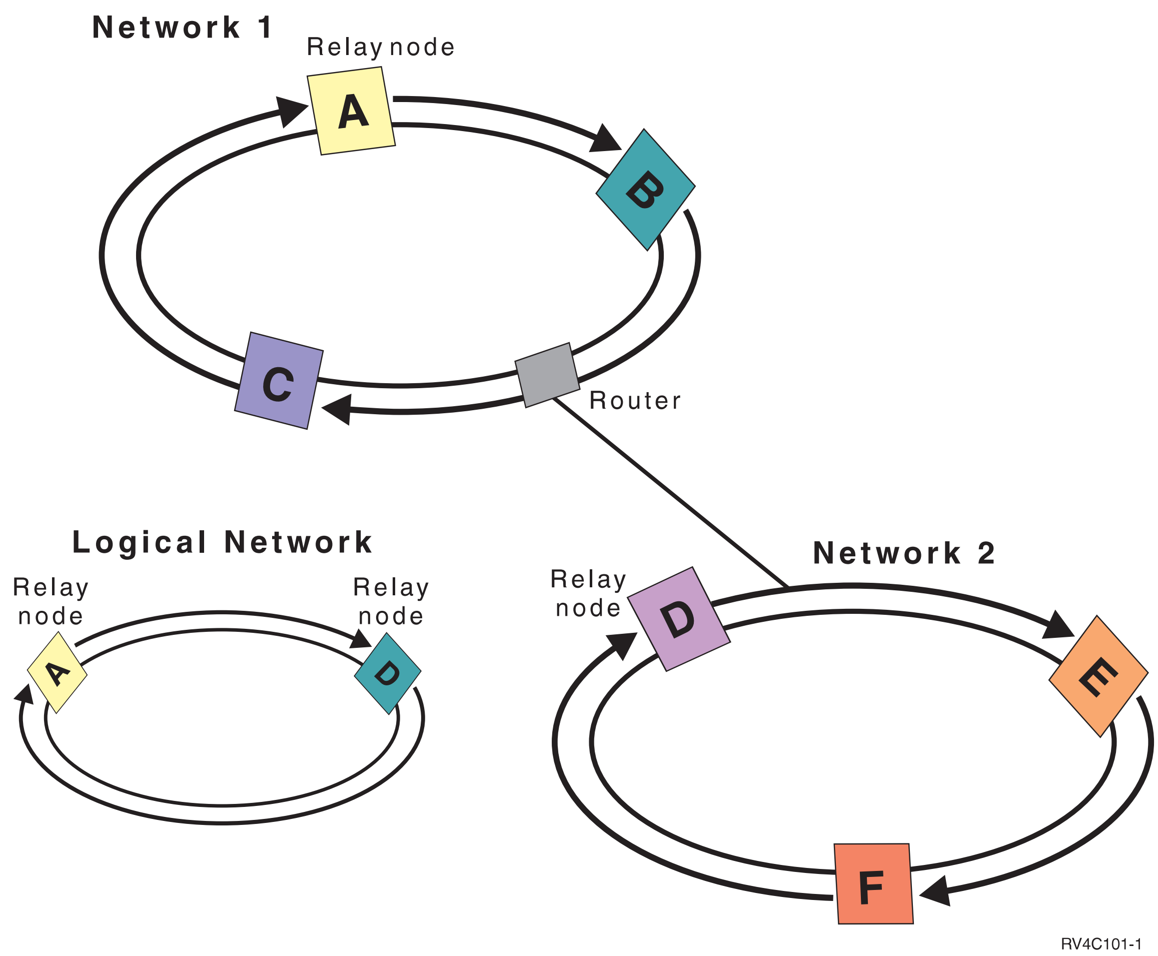
Consider the following examples to understand how heartbeat monitoring works:

Example 1



With the default (or normal) settings, a heartbeat message is sent every 3 seconds from every node in the cluster to its upstream neighbor. For example, if you configure Node A, Node B, and Node C on Network 1, Node A sends a message to Node B, Node B sends a message to Node C, and Node C sends a message to Node A. Node A expects an acknowledgment to the heartbeat from Node B as well as an incoming heartbeat from the downstream Node C. In effect, the heartbeating ring goes both ways. If Node A did not receive a heartbeat from Node C, Node A and Node B continues to send a heartbeat every 3 seconds. If Node C missed four consecutive heartbeats, a heartbeat failure is signaled.

Example 2



Let's add another network to this example to show how routers and relay nodes are used. You configure Node D, Node E, and Node F on Network 2. Network 2 is connected to Network 1 using a router. A router can be another System i product or a router box that directs communications to another router somewhere else. Every local network is assigned a relay node. This relay node is assigned to the node that has the lowest node ID in the network. Node A is assigned as the relay node on Network 1, and Node D is assigned as the relay node on Network 2. A logical network containing Node A and Node D is then created. By using routers and relay nodes, the nodes on these two networks can monitor each other and signal any node failures.

Related concepts

"Managing clusters" on page 112
This topic contains information that covers some of the tasks that involve managing your clusters.

"Cluster performance" on page 127
When changes are made to a cluster, the overhead necessary to manage the cluster can be affected.

Related tasks

"Monitoring cluster status" on page 126
Taking appropriate actions when necessary, the cluster resource services performs basic monitoring of a cluster and its components using the reliable message function and heartbeat monitoring.

Reliable message function

The *reliable message function* of cluster resource services keeps track of each node in a cluster and ensures that all nodes have consistent information about the state of cluster resources.

Reliable messaging uses retry and timeout values that are unique to clustering. These values are preset to values that should suit most environments. However, they can be changed through the Change cluster resource services settings interface. The message retry and timeout values are used to determine how many times a message is sent to a node before a failure or partition situation is signaled. For a local area network (LAN), the amount of time it takes to go through the number of retries before a failure or partition condition is signaled is approximately 45 seconds using the default retry and timeout values. For a remote network, more time is allowed to determine whether a failure or partition condition exists. You can figure approximately 4 minutes and 15 seconds for a remote network.

Related concepts

"Changing cluster resource services settings"
The default values affecting message timeout and retry are set to account for most typical installations. However, it is possible to change these values to more closely match your communications environment.

"Managing clusters" on page 112
This topic contains information that covers some of the tasks that involve managing your clusters.

Related tasks

"Monitoring cluster status" on page 126
Taking appropriate actions when necessary, the cluster resource services performs basic monitoring of a cluster and its components using the reliable message function and heartbeat monitoring.

Changing cluster resource services settings

The default values affecting message timeout and retry are set to account for most typical installations. However, it is possible to change these values to more closely match your communications environment.

The values can be adjusted in one of these ways:

- Set a general performance level that matches your environment
- Set values for specific message tuning parameters for more specific adjustment.

In the first method above, the message traffic is adjusted to one of three communications levels. The normal level is the default and is described in detail in Heartbeat monitoring.

The second method should normally be done only under the advisement of an expert.

The Change Cluster Resource Services (QcstChgClusterResourceServices) API describes details on both methods.

Related concepts

"Heartbeat monitoring" on page 35
Heartbeat monitoring is a cluster resource services function that ensures that each node is active by sending a signal from every node in the cluster to every other node in the cluster to convey that they are still active.

Cluster partition

A *cluster partition* is a subset of the active cluster nodes that results from a communications failure. Members of a partition maintain connectivity with each other.

A cluster partition occurs in a cluster whenever communication is lost between one or more nodes in the cluster and a failure of the lost nodes cannot be confirmed. When a cluster partition condition is detected, cluster resource services limits the types of actions that you can perform on the nodes in the cluster partition. Restricting function during a partition is done so cluster resource services will be able to merge the partitions once the problem that caused it has been fixed.

Certain CRG operations are restricted when a cluster is partitioned. For details on which operations are restricted for each type of partition, see Cluster Resource Group APIs.

If a cluster administrative domain is partitioned, changes will continue to be synchronized among the active nodes in each partition. When the nodes are merged back together again, the cluster administrative domain will propagate all changes made in every partition so that the resources are consistent within the active domain.

Related concepts

"Avoiding a cluster partition" on page 96
The typical network-related cluster partition can best be avoided by configuring redundant communications paths between all nodes in the cluster.

"Partition errors" on page 154
Certain cluster conditions are easily corrected. If a cluster partition has occurred, you can learn how to recover. This topic also tells you how to avoid a cluster partition and gives you an example of how to merge partitions back together.

"Hardware requirements for clusters" on page 91
Any System i model that is capable of running OS/400 V4R4, or later, is compatible for using clustering.

Cluster applications

Application resilience is one of the key elements in a clustered environment. If you are planning to write and use highly available applications in your cluster you should be aware that these applications have specific availability specifications.

By taking advantage of resilient applications in your cluster, an application can be restarted on a different cluster node without requiring you to reconfigure the clients. In addition, the data that is associated with the application will be available after switchover or failover. This means that the application user can experience minimal, or even seamless, interruption while the application and its data switch from the primary node to the backup node. The user does not need to know that the application and data have moved on the back end.

In order to achieve application resiliency in your cluster, applications that meet certain availability specifications must be used. Certain characteristics must be present in the application in order for it to be switchable, and therefore always available to the users of the application in the cluster. See High Availability and Clusters for details on these application traits. Because these requirements exist, you have the following options for using a switchable application in your cluster:

1. **Purchase a cluster-enabled software application**

   Software products that are cluster-enabled meet certain high-availability requirements.
2. **Write or change your own application to make it highly available**

   Independent software vendors and application programmers can customize applications to allow them to be switchable in a System i clustered environment.

Once you have a resilient application, it must be managed within your cluster.

Related concepts

"Resilient applications" on page 23
A *resilient application* is an application that can be restarted on a different cluster node without requiring you to reconfigure the clients.

i5/OS architecture for cluster-enabled applications

Additional end-user value is provided by any application that is highly available, recognizing applications that continue to be available in the event of an outage, planned or unplanned.

i5/OS has provided an application resilience architecture that supports various degrees of highly available application. At the high end of this spectrum applications will be enhanced with integrated functions that demonstrate highly available characteristics and automation of the highly available environment, controlled by cluster management utilities.

These applications have the following characteristics:
- The application can switch over to a backup cluster node when the primary node becomes unavailable.
- The application defines the resilient environment in the Resilient Definition and Status Data Area to enable automatic configuration and activation of the application by a cluster management application.
- The application provides application resilience by means of an application CRG exit program to handle cluster related events, taking advantage of the capabilities of the i5/OS cluster resource services.
- The application provides an application restart function that repositions the user to an application menu screen or beyond.

Applications that demonstrate more stringent availability and restart characteristics have the following characteristics:
- The application provides enhanced application resilience through more robust handling of cluster events (action codes) by the application CRG exit program.
- The application provides a greater level of application restart support. For host-centric applications, the user will be repositioned to a transaction boundary by commitment control or checkpoint functions. For client-centric applications, the user will experience a seamless failover with minimal service interruption.

Related concepts

iSeries High Availability and Clusters

Writing a highly available cluster application

A highly available application is one that can be resilient to a system outage in a clustered environment.

Several levels of application availability are possible:
1. If an application error occurs, the application restarts itself on the same node and corrects any potential cause for error (such as corrupt control data). You can view the application as though it had started for the first time.
2. The application performs some amount of checkpoint-restart processing. You can view the application as if it were close to the point of failure.
3. If a system outage occurs, the application is restarted on a backup server. You can view the application as though it had started for the first time.
4. If a system outage occurs, the application is restarted on a backup server and performs some amount of checkpoint-restart processing across the servers. You can view the application as if it were close to the point of failure.
5. If a system outage occurs, a coordinated failover of both the application and its associated data to another node or nodes in the cluster occurs. You can view the application as though it had started for the first time.
6. If a system outage occurs, a coordinated failover of both the application and its associated data to another node or nodes in the cluster occurs. The application performs some amount of checkpoint-restart processing across the servers. You can view the application as if it were close to the point of failure.

Note: In cases 1 through 4 above, you are responsible for recovering the data.

Making application programs resilient:

Learn how to make application programs resilient.

A resilient application is expected to have the following characteristics:
- The application can be restarted on this node or another node
- The application is accessible to the client through the IP address
- The application is stateless or state information is known

- Data that is associated with the application is available after switchover

The three essential elements that make an application resilient to system outages in a clustered environment are:

The application itself
How tolerant is the application to errors or to system outages, and how transparently can the application restart itself?

The application can handle this through the use of clustering capabilities.

Associated data
When an outage occurs, does it affect the availability of any associated data?

A cluster middleware IBM Business Partner replication product which takes advantage of the clustering capabilities can handle this. Alternatively, the data can be stored in a switchable independent disk pool (switchable independent ASP).

Control capabilities and administration
How easy is it to define the environment that supports the availability of the data and the application?

A third-party cluster management solution that uses the clustering APIs and also combines resilient applications with resilient data can handle this.

Restarting highly available cluster applications:

To restart an application, the application needs to know its state at the time of the failover or switchover.

State information is application specific; therefore, the application must determine what information is needed. Without any state information, the application can be restarted on your PC. However, you will must reestablish your position within the application.

Several methods are available to save application state information for the backup system. Each application needs to determine which method works best for it.

- The application can transfer all state information to the requesting client system. When a switchover or failover occurs, the application uses the stored state on the client to reestablish the state in the new server. This can be accomplished by using the Distribute Information API or Clustered Hash Table APIs.
- The application can replicate state information (such as job information and other control structures that are associated with the application) on a real-time basis. For every change in the structures, the application ships the change over to the backup system.
- The application can store pertinent state information that is associated with its application in the exit program data portion of the cluster resource group for that application. This method assumes that a small amount of state information is required. You can use the Change Cluster Resource Group (QcstChangeClusterResourceGroup) API to do this.
- The application can store state information in a data object that is being replicated to the backup systems along with the application's data.
- The application can store state information in a data object contained in the switchable IASP that also contains the application's data.
- The application can store the state information about the client.
- No state information is saved, and you need to perform the recovery.

Note: The amount of information that is required to be saved is lessened if the application uses some form of checkpoint-restart processing. State information is only saved at predetermined application checkpoints. Restart then takes you back to the last known checkpoint which is similar to how database's commitment control processing works.

Calling a cluster resource group exit program:

The cluster resource group exit program is called during different phases of a cluster environment.

This program establishes the environment necessary resiliency for resources within a cluster. The exit program is optional for a resilient device CRG but is required for the other CRG types. When a cluster resource group exit program is used, it is called on the occurrence of cluster-wide events, including when:

- A node leaves the cluster unexpectedly.
- A node leaves the cluster as a result of the End Cluster Node (QcstEndClusterNode) API or Remove Cluster Node Entry (QcstRemoveClusterNodeEntry) API.
- The cluster is deleted as a result of the Delete Cluster (QcstDeleteCluster) API.
- A node is activated by the Start Cluster Node (QcstStartClusterNode) API.
- Communication with a partitioned node is re-established.

This exit program:

- Runs in a named activation group or the caller's activation group (*CALLER).
- Ignores the restart parameter if the exit program has an unhandled exception or is canceled.
- Provides a cancel handler.

When a cluster resource group API is run, the exit program is called from a separate job with the user profile specified on the Create Cluster Resource Group (QcstCreateClusterResourceGroup) API. The separate job is automatically created by the API when the exit program is called. If the exit program for a data CRG is unsuccessful or ends abnormally, the cluster resource group exit program is called on all active nodes in the recovery domain with an action code of Undo. This action code allows any unfinished activity to be backed out and the original state of the cluster resource group to be recovered.

| Suppose an unsuccessful switchover occurs for a device CRG. After switching back all the devices, if all
| of the devices were varied-on successfully on the original primary node, clustering calls the exit program
| on the original primary node with an action code of Start.

If the exit program for an application CRG is unsuccessful or ends abnormally, cluster resource services will attempt to restart the application if the status of the CRG is active. The cluster resource group exit program is called with an action code of Restart. If the application cannot be restarted in the specified maximum number of attempts, the cluster resource group exit program is called with an action code of Failover. The restart count is reset only when the exit program is called with an action code of start, which can be the result of a start CRG, a failover, or a switchover.

When the cluster resource group is started, the application CRG exit program called on the primary node is not to return control to cluster resource services until the application itself ends or an error occurs. After an application CRG is active, if cluster resource services must notify the application CRG exit program of some event, another instance of the exit program is started in a different job. Any other action code other than Start or Restart is expected to return.

When a cluster resource group exit program is called, it is passed a set of parameters that identify the cluster event being processed, the current state of the cluster resources, and the expected state of the cluster resources.

For complete information about cluster resource group exit programs, including what information is passed to the exit program for each action code, see Cluster Resource Group Exit Program in the Cluster API documentation. Sample source code has been provided in the QUSRTOOL library which can be used as a basis for writing an exit program. See the TCSTAPPEXT member in the QATTSYSC file.

Application CRG considerations

An application cluster resource group manages application resiliency.

Managing application CRG takeover IP addresses:

Manage application CRG takeover IP addresses using cluster resource services. You can also manage them manually.

There are two ways to have the application takeover IP address associated with an application CRG managed. The easiest way, which is the default, is to let cluster resource services manage the takeover IP address. This method will direct cluster resource services to create the takeover IP address on all nodes in the recovery domain, including nodes subsequently added to the recovery domain. When this method is selected, the takeover IP address cannot currently be defined on any node in the recovery domain.

The alternative way is to manage the takeover IP addresses yourself. This method directs cluster resource services to not take any steps to configure the takeover IP address; the user is responsible for configuration. You must add the takeover IP address on all nodes in the recovery domain (except on replicate nodes) before starting the cluster resource group. Any node to be added to the recovery domain of an active CRG must have the takeover IP address configured before being added.

Multiple subnets

It is possible to have the application takeover IP address work across multiple subnets, although the default is to have all recovery domain nodes on the same subnet. See Enabling application switchover for the steps to configure the application takeover IP address when the nodes in the recovery domain span subnets.

Related concepts

"Example: Application cluster resource group failover actions" on page 44
See how an example failover scenario works.

"Creating an application CRG with an active takeover IP address" on page 117
You can specify to allow an active takeover IP address when you create an application CRG. This is only allowed if the user configures the takeover IP address.

Enabling application switchover across subnets:

Clustering in general requires that all cluster nodes in the recovery domain of an application cluster resource group reside on the same LAN (use the same subnet addressing).

The underlying network protocol used to achieve the switching of the configured application takeover IP address from one node in the recovery domain to another is the Address Resolution Protocol (ARP). However, it is possible to extend the recovery domain to include cluster nodes that reside on other LANs separated by commercial routers.

This extension is possible through the use of Virtual IP address support and making use of the Routing Information Protocol (RIP) on the cluster nodes and the commercial routers in the network. See "Enabling application switchover" for more details.

Enabling application switchover:

Cluster resource services supports a user configured takeover IP address when configuring application CRGs.

The following manual configuration steps are required to enable the switchover environment. **This set of instructions must be done on all the nodes in the recovery domain, and repeated for the other nodes in the cluster that will become nodes in the recovery domain for the given application CRG.**

1. Select a takeover IP address to be used by the application CRG.

- To avoid confusion, this address should not overlap with any other existing addresses used by the cluster nodes or routers. For example, if choosing 19.19.19.19, ensure that 19.0.0.0 (19.19.0.0 or ...) are not routes known by the system routing tables.
- Add the takeover interface (for example, 19.19.19.19); create it with a line description of *VIRTUALIP, subnet mask of 255.255.255.255 (host route), Maximum Transmission Unit of 1500 (any number in the range 576-16388), and Autostart of *NO. This takeover address (for example, 19.19.19.19) does must exist as a *VIRTUALIP address before identifying it as an Associated Local Interface in next step. It does not, however, must be active.

2. Associate the intended takeover IP address with one or both of the IP addresses specified to be used by cluster communications when you create the cluster or add a node to the cluster.
   - For example, this means making the 19.19.19.19 takeover address an Associated Local Interface on the IP address for the cluster node on the Ethernet bus to be used locally for clustering. This must be done for each cluster address on each cluster node.

   Note: The cluster addresses will must be ended to accomplish this change under CFGTCP.
3. Create the cluster and create any CRGs. For the application CRG specify `QcstUserCfgsTakeoverIpAddr` for the 'configure takeover IP address' field. Do not start any Application CRGs.
4. Using Configure TCP/IP Applications (option 20) under CFGTCP, then Configure RouteD (option 2), then Change RouteD attributes (option 1), ensure Supply is set to *YES. If not, set to *YES and start or re-start ROUTED (RIP or RIP-2) on each cluster node.
   - NETSTAT option 3 will show ROUTED using a Local Port if currently running. ROUTED must be running and advertising routes (Supply = *YES) on every cluster node in the CRG recovery domain.
5. Ensure that all the commercial routers in the network interconnecting the recovery domain LANs are accepting and advertising host routes for RIP.
   - This is not necessarily the default setting for routers. Language will vary with router manufacturer but under RIP Interfaces, expect to be sending Host Routes and receiving dynamic hosts.
   - This also applies to both the router interfaces pointing to the System i product as well as the router-to-router interfaces

   Note: Do not use a System i product as the router in this scenario. Use a commercial router (IBM or otherwise) that is designed for routing purposes. System i routing cannot be configured to handle this function.
6. You now can manually activate the takeover address on one of the cluster nodes, allow RIP up to 5 minutes to propagate the routes, and ping the takeover address from all nodes in the CRG recovery domain and from selected clients on the LANs who will be using this address.
   - After this verification test, ensure the takeover address is ended again.
   - Clustering will start the address on the specified primary node when the CRGs are started.
7. Start the Application CRGs.
   - The takeover address will now be started by clustering on the specified preferred node and RIP will advertise the routes throughout the recovery domain. RIP may take up to 5 minutes to update routes across the domain. The RIP function is independent from the start CRG function.

Important notes:

- If the above procedure is not followed for all cluster nodes in the application CRG recovery domain, the cluster will hang during the switchover process.
- Even though we do not failover to replica nodes, it is a good idea to perform the procedure on the replica nodes in the event that they may be changed at a later date in time to become a backup.
- If you want to use multiple virtual IP addresses, then each one will require a separate application CRG and a separate IP address with which to be associated. This address may be another logical IP address on the same physical adapter or it may be

another physical adapter altogether. Also, care must be taken to prevent ambiguities in the routing tables. This is best achieved by doing the following:

– Add a *DFTROUTE to the routing table for each virtual IP address.
– This can be done under CFGTCP (option 2).
– Set all parameters, including the next hop, the same to reach the router of choice, but the Preferred binding interface should be set to be the local system IP address associated with the virtual IP address that will be represented by this route.

Example: Application cluster resource group failover actions:

See how an example failover scenario works.

The following happens when a cluster resource group for a resilient application fails over due to exceeding the retry limit or if the job is canceled:

- The cluster resource group exit program is called on all active nodes in the recovery domain for the CRG with an action code of failover. This indicates that cluster resource services is preparing to failover the application's point of access to the first backup.
- Cluster resource services ends the takeover Internet Protocol (IP) connection on the primary node. For more information about the takeover IP address, see Manage application CRG IP addresses.
- Cluster resource services starts the takeover IP address on the first backup (new primary) node.
- Cluster resource services submits a job that calls the cluster resource group exit program only on the new primary node with an action code of Start. This action restarts the application.

The above example shows how one failover scenario works. Other failover scenarios can work differently.

Example: Application exit program:

This code example contains code for a sample application cluster resource group exit program.

You can find this code example in the QUSRTOOL library.

By using the code examples, you agree to the terms of the Code license and disclaimer information.

```
/***************************************************************************/
/*                                                                         */
/* Library:  QUSRTOOL                                                      */
/* File:     QATTSYSC                                                      */
/* Member:   TCSTAPPEXT                                                    */
/* Type:     ILE C                                                         */
/*                                                                         */
/* Description:                                                            */
/* This is an example application CRG exit program which gets called for   */
/* various cluster events or cluster APIs.  The bulk of the logic must     */
/* still be added because that logic is really dependent upon the unique   */
/* things that need to be done for a particular application.               */
/*                                                                         */
/* The intent of this example to to provide a shell which contains the     */
/* basics for building a CRG exit program.  Comments throughout the example*/
/* highlight the kinds of issues that need to be addressed by the real     */
/* exit program implementation.                                            */
/*                                                                         */
/* Every action code that applies to an application CRG is handled in this */
/* example.                                                                */
/*                                                                         */
/* The tcstdtaara.h include is also shipped in the QUSRTOOL library.  See  */
/* the TCSTDTAARA member in the QATTSYSC file.                             */
/*                                                                         */
/* Change log:                                                             */
/* Flag Reason   Ver    Date   User Id   Description                       */
/* ____ ________ ______ ______ ___________________________________________ */
/* ...  D98332   v5r1m0 000509 ROCH     Initial creation.                 */
/* $A1  P9950070 v5r2m0 010710 ROCH     Dataarea fixes                    */
/* $A2  D99055   v5r2m0 010913 ROCH     Added CancelFailover action code  */
/* $A3  D98854   v5r2m0 010913 ROCH     Added VerificationPhase action code*/
/* $A4  P9A10488 v5r3m0 020524 ROCH     Added example code to wait for data*/
/*                                      CRGs on switchover action code    */
/*                                                                        */
/**************************************************************************/


/*------------------------------------------------------------------------*/
/*                                                                        */
/* Header files                                                           */
/*                                                                        */
/*------------------------------------------------------------------------*/
#include              /* Useful when debugging                    */
#include             /* offsetof macro                            */
#include             /* system function                           */
#include             /* String functions                          */
#include             /* Exception handling constants/structures   */
#include               /* Various cluster constants                */
#include           /* Structure of CRG information             */
#include "qusrtool/qattsysc/tcstdtaara" /* QCSTHAAPPI/QCSTHAAPPO data areas*/
#include           /* API to Retrieve contents of a data area  */
#include              /* API error code type definition           */
#include              /* mitime builtin                           */
#include           /* waittime builtin                         */


/*------------------------------------------------------------------------*/
/*                                                                        */
/* Constants                                                              */
/*                                                                        */
/*------------------------------------------------------------------------*/
#define UnknownRole -999
#define DependCrgDataArea "QCSTHAAPPO"
#define ApplCrgDataArea "QCSTHAAPPI"
#define Nulls 0x00000000000000000000


/*------------------------------------------------------------------------*/
/*                                                                        */
/* The following constants are used in the checkDependCrgDataArea()       */
/* function.  The first defines how long to sleep before checking the data */
/* area.  The second defines that maximum time to wait for the data area  */
/* to become ready before failing to start the application when the Start  */
/* CRG function is being run.  The third defines the maximum wait time for */
/* the Initiate Switchover or failover functions.                         */
/*                                                                        */
/*------------------------------------------------------------------------*/
#define WaitSecondsIncrement 30
#define MaxStartCrgWaitSeconds 0
#define MaxWaitSeconds 900

/*------------------------------------------------------------------------*/
/*                                                                        */
/* As this exit program is updated to handle new action codes, change the  */
/* define below to the value of the highest numbered action code that is   */
/* handled.                                                               */
/*                                                                        */
/*------------------------------------------------------------------------*/
#define MaxAc 21

/*------------------------------------------------------------------------*/
/*                                                                        */
/* If the exit program data in the CRG has a particular structure to it,   */
/* include the header file for that structure definition and change the    */
/* define below to use that structure name rather than char.              */
/*                                                                        */
/*------------------------------------------------------------------------*/
#define EpData char

/*------------------------------------------------------------------------*/
/*                                                                        */
/* Change the following define to the library the application resides in */
/* and thus where the QCSTHAAPPO and QCSTHAAPPI data areas will be found. */
/*                                                                        */
/*------------------------------------------------------------------------*/
#define ApplLib "QGPL"


/*------------------------------------------------------------------------*/
/*                                                                        */
/* Prototypes for internal functions.                                     */
/*                                                                        */
/*------------------------------------------------------------------------*/
static int getMyRole(Qcst_EXTP0100_t *, int, int);
#pragma argopt(getMyRole)
static int doAction(int, int, int, Qcst_EXTP0100_t *, EpData *);
#pragma argopt(doAction)
static int createCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int startCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int restartCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int endCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int verifyPhase(int, int, Qcst_EXTP0100_t *, EpData *);
static int deleteCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int memberIsJoining(int, int, Qcst_EXTP0100_t *, EpData *);
static int memberIsLeaving(int, int, Qcst_EXTP0100_t *, EpData *);
static int switchPrimary(int, int, Qcst_EXTP0100_t *, EpData *);
static int addNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int rmvNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int chgCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int deleteCrgWithCmd(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoPriorAction(int, int, Qcst_EXTP0100_t *, EpData *);
static int endNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int chgNodeStatus(int, int, Qcst_EXTP0100_t *, EpData *);
static int cancelFailover(int, int, Qcst_EXTP0100_t *, EpData *);
static int newActionCode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoCreateCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoStartCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoEndCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoMemberIsJoining(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoMemberIsLeaving(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoSwitchPrimary(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoAddNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoRmvNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoChgCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoCancelFailover(int, int, Qcst_EXTP0100_t *, EpData *);
static void bldDataAreaName(char *, char *, char *);
#pragma argopt(bldDataAreaName)
static int checkDependCrgDataArea(unsigned int);
#pragma argopt(checkDependCrgDataArea)
static void setApplCrgDataArea(char);
#pragma argopt(setApplCrgDataArea)
static void cancelHandler(_CNL_Hndlr_Parms_T *);
static void unexpectedExceptionHandler(_INTRPT_Hndlr_Parms_T *);
static void endApplication(unsigned int, int, int, Qcst_EXTP0100_t *, EpData *);
#pragma argopt(endApplication)

/*------------------------------------------------------------------------*/
/*                                                                        */
/* Some debug routines                                                    */
/*                                                                        */
/*------------------------------------------------------------------------*/
```

```
static void printParms(int, int, int, Qcst_EXTP0100_t *, EpData *);
static void printActionCode(unsigned int);
static void printCrgStatus(int);
static void printRcvyDomain(char *,
                            unsigned int,
                            Qcst_Rcvy_Domain_Array1_t *);
static void printStr(char *, char *, unsigned int);


/*-------------------------------------------------------------------------*/
/*                                                                         */
/* Type definitions                                                        */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*                                                                         */
/* This structure defines data that will be passed to the exception and    */
/* cancel handlers.  Extend it with information unique to your application.*/
/*                                                                         */
/*-------------------------------------------------------------------------*/
typedef struct {
  int *retCode;             /* Pointer to return code                      */
  EpData *epData;           /* Exit program data from the CRG              */
  Qcst_EXTP0100_t *crgData; /* CRG data                                    */
  unsigned int actionCode;  /* The action code                             */
  int role;                 /* This node's recovery domain role            */
  int priorRole;            /* This node's prior recovery domainrole       */
} volatile HandlerDataT;


/*-------------------------------------------------------------------------*/
/*                                                                         */
/* Function pointer array for handling action codes.  When the exit program*/
/* is updated to handle new action codes, add the new function names to    */
/* this function pointer array.                                            */
/*                                                                         */
/*-------------------------------------------------------------------------*/
static int (*fcn[MaxAc+1]) (int role,
                            int priorRole,
                            Qcst_EXTP0100_t *crgData,
                            EpData *epData) = {
  newActionCode,   /* 0 - currently reserved */
  createCrg,       /* 1  */
  startCrg,        /* 2  */
  restartCrg,      /* 3  */
  endCrg,          /* 4  */
  verifyPhase,     /* 5 - currently reserved */
  newActionCode,   /* 6 - currently reserved */
  deleteCrg,       /* 7  */
  memberIsJoining, /* 8  */
  memberIsLeaving, /* 9  */
  switchPrimary,   /* 10 */
  addNode,         /* 11 */
  rmvNode,         /* 12 */
  chgCrg,          /* 13 */
  deleteCrgWithCmd,/* 14 */
  undoPriorAction, /* 15 */
  endNode,         /* 16 */
  newActionCode,   /* 17 - applies only to a device CRG */
  newActionCode,   /* 18 - applies only to a device CRG */
  newActionCode,   /* 19 - applies only to a device CRG */
  chgNodeStatus,   /* 20 */
  cancelFailover   /* 21 */
};
```

```
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* Function pointer array for handling prior action codes when called with */
/* the Undo action code.  When the exit program is updated to handle       */
/* Undo for new action codes, add the new function names to this function  */
/* pointer array.                                                          */
/*                                                                         */
/*-------------------------------------------------------------------------*/
static int (*undoFcn[MaxAc+1]) (int role,
                                int priorRole,
                                Qcst_EXTP0100_t *crgData,
                                EpData *epData) = {
  newActionCode,       /* 0 - currently reserved */
  undoCreateCrg,       /* 1  */
  undoStartCrg,        /* 2  */
  newActionCode,       /* 3  */
  undoEndCrg,          /* 4  */
  newActionCode,       /* 5 - no undo for this action code */
  newActionCode,       /* 6 - currently reserved */
  newActionCode,       /* 7  */
  undoMemberIsJoining, /* 8  */
  undoMemberIsLeaving, /* 9  */
  undoSwitchPrimary,   /* 10 */
  undoAddNode,         /* 11 */
  undoRmvNode,         /* 12 */
  undoChgCrg,          /* 13 */
  newActionCode,       /* 14 */
  newActionCode,       /* 15 */
  newActionCode,       /* 16 */
  newActionCode,       /* 17 - applies only to a device CRG */
  newActionCode,       /* 18 - applies only to a device CRG */
  newActionCode,       /* 19 - applies only to a device CRG */
  newActionCode,       /* 20 */
  undoCancelFailover   /* 21 */
};



/***************************************************************************/
/*                                                                         */
/* This is the entry point for the exit program.                           */
/*                                                                         */
/***************************************************************************/
void main(int argc, char *argv[]) {

  HandlerDataT hdlData;


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* Take each of the arguments passed in the argv array and castit to   */
  /* the correct data type.                                              */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  int *retCode    = (int *)argv[1];
  unsigned int *actionCode = (unsigned int *)argv[2];
  EpData *epData           = (EpData *)argv[3];
  Qcst_EXTP0100_t *crgData = (Qcst_EXTP0100_t *)argv[4];
  char *formatName         = (char *)argv[5];


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* Ensure the format of the data being passed is what we areexpecting. */
  /* If not, a change has been made and this exit program needs tobe     */
  /* updated to accomodate the change.  Add appropriate errorlogging for */
  /* your application design.                                         */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  if (0 != memcmp(formatName, "EXTP0100", 8))
    abort();


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set up the data that will be passed to the exception andcancel   */
  /* handlers.                                                        */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  hdlData.retCode    = retCode;
  hdlData.epData     = epData;
  hdlData.crgData    = crgData;
  hdlData.actionCode = *actionCode;
  hdlData.role       = UnknownRole;
  hdlData.priorRole  = UnknownRole;
  _VBDY();  /* force changed variables to home storage location       */


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Enable an exception handler for any and all exceptions.          */
  /*                                                                  */

/*--------------------------------------------------------------------*/
#pragma exception_handler(unexpectedExceptionHandler, hdlData, \
                          _C1_ALL, _C2_ALL, _CTLA_INVOKE )


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Enable a cancel handler to recover if this job is canceled.     */
  /*                                                                  */

/*--------------------------------------------------------------------*/
#pragma cancel_handler(cancelHandler, hdlData)


/*--------------------------------------------------------------------*/
  /*                                                                   */
  /* Extract the role and prior role of the node this exit program is */
  /* running on.  If the cluster API or event changes the recovery domain */
  /* (node role or membership status), the new recovery domain's offset is */
  /* passed in Offset_Rcvy_Domain_Array and the offset of the recovery  */
  /* domain as it looked prior to the API or cluster event is passed in */
  /* Offset_Prior_Rcvy_Domain_Array.  If the recovery domain isn't changed,*/
  /* only Offset_Rcvy_Domain_Array can be used to address the recovery  */
  /* domain.                                                            */
  /*                                                                    */

/*--------------------------------------------------------------------*/
  hdlData.role = getMyRole(crgData,
                           crgData->Offset_Rcvy_Domain_Array,
                           crgData->Number_Nodes_Rcvy_Domain);
  if (crgData->Offset_Prior_Rcvy_Domain_Array)
    hdlData.priorRole =
                 getMyRole(crgData,

crgData->Offset_Prior_Rcvy_Domain_Array,

crgData->Number_Nodes_Prior_Rcvy_Domain);
  else
```

```
    hdlData.priorRole = hdlData.role;
  _VBDY();  /* force changed variables to home storage location           */


/*----------------------------------------------------------------------*/
  /*                                                                    */
  /* Enable the following to print out debug information.               */
  /*                                                                    */

/*----------------------------------------------------------------------*/
  /*
  printParms(*actionCode, hdlData.role, hdlData.priorRole, crgData,
epData);
  */


/*----------------------------------------------------------------------*/
  /*                                                                      */
  /* Do the correct thing based upon the action code.  The return code     */
  /* is set to the function result of doAction().                          */
  /*                                                                      */

/*----------------------------------------------------------------------*/
  *retCode = doAction(*actionCode,
                      hdlData.role,
                      hdlData.priorRole,
                      crgData,
                      epData);


/*----------------------------------------------------------------------*/
  /*                                                                      */
  /* The exit program job will end when control returns to the operating   */
  /* system at this point.                                                 */
  /*                                                                      */

/*----------------------------------------------------------------------*/
  return;

#pragma disable_handler  /* unexpectedExceptionHandler                    */
#pragma disable_handler  /* cancelHandler                                 */
}  /* end main()                                                          */


/************************************************************************/
/*                                                                        */
/* Get the role of this particular node from one of the views of the      */
/* recovery domain.                                                       */
/*                                                                        */
/* APIs and cluster events which pass the updated and prior recovery domain*/
/* to the exit program are:                                               */
/* QcstAddNodeToRcvyDomain                                                */
/* QcstChangeClusterNodeEntry                                             */
/* QcstChangeClusterResourceGroup                                         */
/* QcstEndClusterNode (ending node does not get the prior domain)         */
/* QcstInitiateSwitchOver                                                 */
/* QcstRemoveClusterNodeEntry (removed node does not get the prior domain) */
/* QcstRemoveNodeFromRcvyDomain                                           */
/* QcstStartClusterResourceGroup (only if inactive backup nodes are       */
/*                               reordered)                               */
/* a failure causing failover                                             */
/* a node rejoining the cluster                                           */
/* cluster partitions merging                                             */
/*                                                                        */
/* All other APIs pass only the updated recovery domain.                  */
/*                                                                        */
/************************************************************************/
```

```
static int getMyRole(Qcst_EXTP0100_t *crgData, int offset, int
count) {

  Qcst_Rcvy_Domain_Array1_t *nodeData;
  unsigned int iter = 0;


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* Under some circumstances, the operating system may not be able to     */
  /* determine the ID of this node and passes *NONE.  An example of such a */
  /* circumstance is when cluster resource services is not active on a     */
  /* node and the DLTCRG CL command is used.                               */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  if (0 == memcmp(crgData->This_Nodes_ID, QcstNone,
sizeof(Qcst_Node_Id_t)))
    return UnknownRole;


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* Compute a pointer to the first element of the recovery domain array.  */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  nodeData = (Qcst_Rcvy_Domain_Array1_t *)((char *)crgData +
offset);


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* Find my node in the recovery domain array.  I will not be in the      */
  /* prior recovery domain if I am being added by the Add Node to Recovery */
  /* Domain API.                                                           */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  while (  0 != memcmp(crgData->This_Nodes_ID,
                       nodeData->Node_ID,
                       sizeof(Qcst_Node_Id_t))
         &&
           iter &lt; count
        ) {
    nodeData++;
    iter++;
  }

  if (iter &lt; count)
    return nodeData->Node_Role;
  else
    return UnknownRole;
}  /* end getMyRole()                                                      */


/***************************************************************************/
/*                                                                         */
/* Call the correct function based upon the cluster action code.  The      */
/* doAction() function was split out from main() in order to clarify the   */
/* example.  See the function prologues for each called function for       */
/* information about a particular cluster action.                          */
/*                                                                         */
/* Each action code is split out into a separate function only to help     */
/* clarify this example.  For a particular exit program, some action codes */
/* may perform the same function in which case multiple action codes could */
/* be handled by the same function.                                        */
/*                                                                        */
/**************************************************************************/
static int doAction(int actionCode,
                    int role,
                    int priorRole,
                    Qcst_EXTP0100_t *crgData,
                    EpData *epData) {


/*--------------------------------------------------------------------*/
  /*                                                                    */
  /* For action codes this exit program knows about, call a function to */
  /* do the work for that action code.                                  */
  /*                                                                    */

/*--------------------------------------------------------------------*/

  if (actionCode &lt;= MaxAc )
    return (*fcn[actionCode]) (role, priorRole, crgData, epData);
  else

/*------------------------------------------------------------------*/
    /*                                                                  */
    /* IBM has defined a new action code in a new operating system release */
    /* and this exit program has not yet been updated to handle it.  Take a*/
    /* default action for now.                                          */
    /*                                                                  */

/*------------------------------------------------------------------*/
    return newActionCode(role, priorRole, crgData, epData);
}  /* end doAction()                                                      */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcInitialize                                      */
/*                                                                        */
/* The QcstCreateClusterResourceGroup API was called.  A new cluster      */
/* resource group object is being created.                                */
/*                                                                        */
/* Things to consider:                                                    */
/*   - Check that the application program and all associated objects are on*/
/*     the primary and backup nodes.  If the objects are not there,       */
/*     consider sending error/warning messages or return a failure return  */
/*     code.                                                              */
/*   - Check that required data or device CRGs are on all nodes in the     */
/*     recovery domain.                                                   */
/*   - Perform any necessary setup that is required to run the       */
/*     the application on the primary or backup nodes.                    */
/*   - If this CRG is enabled to use the QcstDistributeInformation API,    */
/*     the user queue needed by that API could be created at this time.   */
/*                                                                        */
/**************************************************************************/
static int createCrg(int role,
                     int doesNotApply,
                     Qcst_EXTP0100_t *crgData,
                     EpData *epData) {

  return QcstSuccessful;
}  /* end createCrg()                                                     */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcStart                                           */
/*                                                                        */
/* The QcstStartClusterResourceGroup API was called.  A cluster resource   */
/* group is being started.                                             */
/* The QcstInitiateSwitchOver API was called and this is the second action */
/* code being passed to the exit program.                              */
/* The fail over event occurred and this is the second action code being   */
/* passed to the exit program.                                         */
/*                                                                     */
/* A maximum wait time is used when checking to see if all dependent CRGs  */
/* are active.  This is a short time if the CRG is being started because of*/
/* the QcstStartClusterResourceGroup API.  It is a longer time if it is    */
/* because of a failover or switchover.  When failover or switchover are   */
/* being done, it make take a while for data or device CRGs to become      */
/* ready so the wait time is long.  If the Start CRG API is being used, the*/
/* dependent CRGs should already be started or some error occurred, the    */
/* CRGs were started out of order, etc. and there is no need for a long    */
/* wait.                                                               */
/*                                                                     */
/* Things to consider:                                                 */
/*   - If this node's role is primary, the application should be started.  */
/*     This exit program should either call the application so that it runs*/
/*     in this same job or it should monitor any job started by this       */
/*     exit program so the exit program knows when the application job     */
/*     ends.  By far, the simplest approach is run the application in this */
/*     job by calling it.                                              */
/*     Cluster Resource Services is not expecting this exit program to     */
/*     return until the application finishes running.                  */
/*   - If necessary, start any associated subsystems, server jobs, etc.    */
/*   - Ensure that required data CRGs have a status of active on all nodes */
/*     in the recovery domain.                                         */
/*                                                                     */
/***********************************************************************/
static int startCrg(int role,
                    int doesNotApply,
                    Qcst_EXTP0100_t *crgData,
                    EpData *epData) {

  unsigned int maxWaitTime;

  /* Start the application if this node is the primary                 */
  if (role == QcstPrimaryNodeRole) {

/*-------------------------------------------------------------------*/
    /*                                                                 */
    /* Determine if all CRGs that this application CRG is dependent upon   */
    /* are ready.  If the check fails, return from the Start action code.  */
    /* Cluster Resource Services will change the state of the CRG to       */
    /* Inactive.                                                       */
    /*                                                                 */

/*-------------------------------------------------------------------*/
    if (crgData->Cluster_Resource_Group_Status ==
QcstCrgStartCrgPending)
      maxWaitTime = MaxStartCrgWaitSeconds;
    else
      maxWaitTime = MaxWaitSeconds;
    if (QcstSuccessful != checkDependCrgDataArea(maxWaitTime))
      return QcstSuccessful;



/*-------------------------------------------------------------------*/
    /*                                                                 */
    /* Just before starting the application, update the data area to       */
    /* indicate the application is running.                            */
    /*                                                                 */

/*-------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Running);


/*-------------------------------------------------------------------*/
    /*                                                                  */
    /* Add logic to call application here.  It is expected that control    */
    /* will not return until something causes the application to end: a    */
    /* normal return from the exit program, the job is canceled, or an    */
    /* unhandled exception occurs.  See the cancelHandler() function for   */
    /* some common ways this job could be canceled.                      */
    /*                                                                  */

/*-------------------------------------------------------------------*/




/*-------------------------------------------------------------------*/
    /*                                                                  */
    /* After the application has ended normally, update the data area to   */
    /* indicate the application is no longer running.                   */
    /*                                                                  */

/*-------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }
  else

/*-------------------------------------------------------------------*/
    /*                                                                  */
    /* On backup or replicate nodes, mark the status of the application in */
    /* the data area as not running.                                    */
    /*                                                                  */

/*-------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


  return QcstSuccessful;
}  /* end startCrg()
       */


/***************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcRestart                                        */
/*                                                                       */
/* The previous call of the exit program failed and set the return   */
/* code to QcstFailWithRestart or it failed due to an exception and the    */
/* exception was allowed to percolate up the call stack.  In either  */
/* case, the maximum number of times for restarting the exit program has   */
/* not been reached yet.                                                 */
/*                                                                       */
/* This action code is passed only to application CRG exit programs which  */
/* had been called with the Start action code.                           */
/*                                                                       */
/***************************************************************************/
static int restartCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {


/*---------------------------------------------------------------------*/
  /*                                                                    */
  /* Perform any unique logic that may be necessary when restarting the    */
  /* application after a failure and then call the startCrg() function to  */
  /* do the start functions.                                            */
```

```
|    /*                                                                    */
|
|  /*--------------------------------------------------------------------*/
|
|
|    return startCrg(role, doesNotApply, crgData, epData);
|  }  /* end restartCrg()                                                   */
|
|
|  /**********************************************************************/
|  /*                                                                    */
|  /* Action code = QcstCrgAcEnd                                         */
|  /*                                                                    */
|  /* The end action code is used for one of the following reasons:     */
|  /*   - The QcstEndClusterResourceGroup API was called.                */
|  /*   - The cluster has become partitioned and this node is in the secondary*/
|  /*     partition.  The End action code is used regardless of whether the   */
|  /*     CRG was active or inactive.  Action code dependent data of          */
|  /*     QcstPartitionFailure will also be passed.                           */
|  /*   - The application ended.   Action code dependent data of              */
|  /*     QcstResourceEnd will also be passed.  All nodes in the recovery     */
|  /*     domain will see the same action code (including the primary).      */
|  /*   - The CRG job has been canceled.  The exit program on this node will */
|  /*     be called with the End action code.  QcstMemberFailure will be      */
|  /*     passed as action code dependent data.                               */
|  /*                                                                    */
|  /*                                                                    */
|  /*                                                                    */
|  /* Things to consider:                                                */
|  /*   - If the CRG is active, the job running the application is canceled  */
|  /*     and the IP takeover address is ended AFTER the exit program is      */
|  /*     called.                                                             */
|  /*   - If subsystems or server jobs were started as a result of the        */
|  /*     QcstCrgAcStart action code, end them here or consolidate all logic  */
|  /*     to end the application in the cancelHandler() since it will be      */
|  /*     invoked for all Cluster Resource Services APIs which must end the   */
|  /*     application on the current primary.                                 */
|  /*                                                                    */
|  /**********************************************************************/
|  static int endCrg(int role,
|                    int priorRole,
|                    Qcst_EXTP0100_t *crgData,
|                    EpData *epData) {
|
|
|  /*--------------------------------------------------------------------*/
|    /*                                                                    */
|    /* End the application if it is running on this node.                 */
|    /*                                                                    */
|
|  /*--------------------------------------------------------------------*/
|    endApplication(QcstCrgAcRemoveNode, role, priorRole, crgData,
|  epData);
|
|    return QcstSuccessful;
|  }  /* end endCrg()                                                       */
|
|
|  /**********************************************************************/
|  /*                                                                    */
|  /* Action code = QcstCrgAcVerificationPhase                           */
|  /*                                                                    */
|  /* The verification phase action code is used to allow the exit program to */
|  /* do some verification before proceeding with the requested function      */
|  /* identified by the action code depended data. If the exit program        */
|  /* determines that the requested function cannot proceed it should return  */
|  /* QcstFailWithOutRestart.                                                 */
```

```
/*                                                                        */
/*                                                                        */
/* NOTE: The exit program will NOT be called with Undo action code.       */
/*                                                                        */
/**************************************************************************/
static int verifyPhase(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {


/*----------------------------------------------------------------------*/
  /*                                                                      */
  /* Do verification                                                      */
  /*                                                                      */

/*----------------------------------------------------------------------*/
  if (crgData->Action_Code_Dependent_Data == QcstDltCrg) {
        /* do verification  */
        /* if ( fail ) */
          /* return QcstFailWithOutRestart */
  }

  return QcstSuccessful;
}  /* end verifyPhase()                                                   */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcDelete                                          */
/*                                                                        */
/* The QcstDeleteClusterResourceGroup or QcstDeleteCluster API was called. */
/* A cluster resource group is being deleted while Cluster Resource       */
/* Services is active.                                                    */
/* If the QcstDeleteCluster API was used, action code dependent data of    */
/* QcstDltCluster is passed.                                              */
/* If the QcstDeleteCluster API was used and the CRG is active, the exit   */
/* program job which is still active for the Start action code is canceled*/
/* after the Delete action code is processed.                             */
/*                                                                        */
/* Things to consider:                                                    */
/*   - Delete application programs and objects from nodes where they are   */
/*     no longer needed such as backup nodes.  Care needs to be exercised  */
/*     when deleting application objects just because a CRG is being       */
/*     deleted since a particular scenario may want to leave the           */
/*     application objects on all nodes.                                  */
/*                                                                        */
/**************************************************************************/
static int deleteCrg(int role,
                     int doesNotApply,
                     Qcst_EXTP0100_t *crgData,
                     EpData *epData) {

  return QcstSuccessful;
}  /* end deleteCrg()
     */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcReJoin                                          */
/*                                                                        */
/* One of three things is occurring-                                      */
/*  1. The problem which caused the cluster to become partitioned has been */
/*     corrected and the 2 partitions are merging back together to become  */
/*     a single cluster.  Action code dependent data of QcstMerge will be  */
/*     passed.                                                            */
/*  2. A node which either previously failed or which was ended has had    */
/*     cluster resource services started again and the node is joining the */
/*     cluster.  Action code dependent data of QcstJoin will be passed.    */
/*  3. The CRG job on a particular node which may have been canceled or   */
/*     ended has been restarted.  Action code dependent data of QcstJoin   */
/*     will be passed.                                                     */
/*                                                                         */
/* Things to consider:                                                     */
/*   - If the application replicates application state information to other*/
/*     nodes when the application is running, this state information will  */
/*     need to be resynchronized with the joining nodes if the CRG is      */
/*     active.                                                             */
/*   - Check for missing application objects on the joining nodes.         */
/*   - Ensure the required data CRGs are on the joining nodes.             */
/*   - If the application CRG is active, ensure the required data CRGs are */
/*     active.                                                             */
/*                                                                         */
/***************************************************************************/
static int memberIsJoining(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*---------------------------------------------------------------------*/
  /*                                                                     */
  /* Ensure the data area status on this node starts out indicating      */
  /* the application is not running if this node is not the primary.    */
  /*                                                                     */

/*---------------------------------------------------------------------*/
  if (role != QcstPrimaryNodeRole) {
    setApplCrgDataArea(Appl_Ended);
  }


/*-----------------------------------------------------------------------*/
  /*                                                                       */
  /* If a single node is rejoining the cluster, you may do a certain set of*/
  /* actions.  Whereas if the nodes in a cluster which became partitioned  */
  /* are merging back together, you may have a different set of actions.   */
  /*                                                                       */

/*-----------------------------------------------------------------------*/
  if (crgData->Action_Code_Dependent_Data == QcstJoin) {
    /* Do actions for a node joining.                                      */
  }
  else {
    /* Do actions for partitions merging.                                  */
  }

  return QcstSuccessful;
}  /* end memberIsJoining()                                                */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcFailover                                         */
/*                                                                         */
/* Cluster resource services on a particular node(s) has failed or ended   */
/* for this cluster resource group.  The Failover action code is passed    */
/* regardless of whether the CRG is active or inactive.  Failover can      */
/* happen for a number of reasons:                                         */
/*                                                                         */
/*   - an operator canceled the CRG job on a node.  Action code dependent */
/*     data of QcstMemberFailure will be passed.                           */
/*   - cluster resource services was ended on the node (for example, the   */
/*     QSYSWRK subsystem was ended with CRS still active).  Action code    */
/*     dependent data of QcstNodeFailure will be passed.                   */
/*   - the application for an application CRG has failed on the primary    */
/*     node and could not be restarted there.  The CRG is Active.          */
/*     Action code dependent data of QcstApplFailure will be passed.       */
/*   - the node failed (such as a power failure).  Action code dependent   */
/*     data of QcstNodeFailure will be passed.                             */
/*   - The cluster has become partitioned due to some communication failure*/
/*     such as a communication line or LAN failure.  The Failover action   */
/*     code is passed to recovery domain nodes in the majority partition.  */
/*     Nodes in the minority partition see the End action code.  Action    */
/*     code dependent data of QcstPartitionFailure will be passed.         */
/*   - A node in the CRG's recovery domain is being ended with the         */
/*     QcstEndClusterNode API.  The node being ended will see the End Node */
/*     action code.  All other nodes in the recovery domain will see the   */
/*     Failover action code.  Action code dependent data of QcstEndNode    */
/*     will be passed for the Failover action code.                        */
/*   - An active recovery domain node for an active CRG is being removed   */
/*     from the cluster with the QcstRemoveClusterNodeEntry API.   Action  */
/*     code dependent data of QcstRemoveNode will be passed.  If an        */
/*     inactive node is removed for an active CRG, or if the CRG is        */
/*     inactive, an action code of Remove Node is passed.                  */
/*                                                                         */
/* The exit program is called regardless of whether or not the CRG is     */
/* active.  The exit program may have nothing to do if the CRG is not      */
/* active.                                                                 */
/*                                                                         */
/* If the CRG is active and the leaving member was the primary node,       */
/* perform the functions necessary for failover to a new primary.          */
/*                                                                         */
/* The Action_Code_Dependent_Data field can be used to determine if:       */
/*  - the failure was due to a problem that caused the cluster to become   */
/*     partitioned (all CRGs which had the partitioned nodes in the        */
/*     recovery domain are affected)                                       */
/*  - a node failed or had cluster resource services ended on the node (all*/
/*    CRGs which had the failed/ended node in the recovery  domain are     */
/*    affected)                                                            */
/*  - only a single CRG was affected (for example a single CRG job was     */
/*    canceled on a node or a single application failed)                  */
/*                                                                         */
/*                                                                         */
/* Things to consider:                                                     */
/*   - Prepare the new primary node so the application can be started.     */
/*   - The application should NOT be started at this time.  The exit       */
/*     program will be called again with the QcstCrgAcStart action code if */
/*     the CRG was active when the failure occurred.                       */
/*   - If the application CRG is active, ensure the required data CRGs are */
/*     active.                                                             */
/*                                                                         */
/***************************************************************************/
static int memberIsLeaving(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*-----------------------------------------------------------------------*/
  /*                                                                       */
  /* If the CRG is active, perform failover.  Otherwise, nothing to do.    */
  /*                                                                       */

/*-----------------------------------------------------------------------*/
  if (crgData->Original_Cluster_Res_Grp_Stat == QcstCrgActive) {


/*---------------------------------------------------------------------*/
    /*                                                                     */
```

```
      /* The CRG is active.  Determine if my role has changed and I am now   */
      /* the new primary.                                                    */
      /*                                                                     */

  /*--------------------------------------------------------------------*/

      if (priorRole != role && role == QcstPrimaryNodeRole) {


  /*------------------------------------------------------------------*/
        /*                                                                   */
        /* I was not the primary but am now.  Do failover actions but don't  */
        /* start the application at this time because this exit program will */
        /* be called again with the Start action code.                       */
        /*                                                                   */

  /*------------------------------------------------------------------*/


  /*------------------------------------------------------------------*/
        /*                                                                   */
        /* Ensure the data area status on this node starts out indicating    */
        /* the application is not running.                                   */
        /*                                                                   */

  /*------------------------------------------------------------------*/
        setApplCrgDataArea(Appl_Ended);


  /*------------------------------------------------------------------*/
        /*                                                                   */
        /* If the application has no actions to do on the Start action code  */
        /* and will become active as soon as the takeover IP address is      */
        /* activated, then this code should be uncommented.  This code will  */
        /* determine if all CRGs that this application CRG is dependent upon */
        /* are ready.  If this check fails, return failure from the action   */
        /* code.                                                             */
        /*                                                                   */

  /*------------------------------------------------------------------*/
  /*      if (QcstSuccessful != checkDependCrgDataArea(MaxWaitSeconds))  */
  /*         return QcstFailWithOutRestart;                              */

      }
    }

    return QcstSuccessful;
  }  /* end memberIsLeaving()                                              */


  /***********************************************************************/
  /*                                                                     */
  /* Action code = QcstCrgAcSwitchover                                   */
  /*                                                                     */
  /* The QcstInitiateSwitchOver API was called.  The first backup node in */
  /* the cluster resource group's recovery domain is taking over as the   */
  /* primary node and the current primary node is being made the last backup.*/
  /*                                                                     */
  /* Things to consider:                                                 */
  /*   - Prepare the new primary node so the application can be started.  */
  /*   - The application should NOT be started at this time.  The exit   */
  /*     program will be called again with the QcstCrgAcStart action code. */
  /*   - The job running the application is canceled and the IP takeover  */
  /*     address is ended prior to the exit program being called on the   */
  /*     current primary.                                                */
  /*   - Ensure required data or device CRGs have switched over and are   */
  /*     active.                                                         */
```

```
/*                                                                        */
/**************************************************************************/
static int switchPrimary(int role,
                         int priorRole,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {


/*----------------------------------------------------------------------*/
  /*                                                                      */
  /* See if I am the old primary.                                         */
  /*                                                                      */

/*----------------------------------------------------------------------*/
  if (priorRole == QcstPrimaryNodeRole) {

/*--------------------------------------------------------------------*/
    /*                                                                    */
    /* Do what ever needs to be done to cleanup the old primary before the */
    /* switch.  Remember that that job which was running the exit program  */
    /* which started the application was canceled already.               */
    /*                                                                    */
    /* One example may be to clean up any processes holding locks on the   */
    /* database.  This may have been done by the application cancel        */
    /* handler if one was invoked.                                         */

/*--------------------------------------------------------------------*/
  }


/*----------------------------------------------------------------------*/
  /*                                                                      */
  /* I'm not the old primary.  See if I'm the new primary.                */
  /*                                                                      */

/*----------------------------------------------------------------------*/
  else if (role == QcstPrimaryNodeRole) {

/*--------------------------------------------------------------------*/
    /*                                                                    */
    /* Do what ever needs to be done on the new primary before the        */
    /* application is started with the QcstCrgAcStart action code.        */
    /*                                                                    */

/*--------------------------------------------------------------------*/


/*--------------------------------------------------------------------*/
    /*                                                                    */
    /* Ensure the data area status on this nodes starts out indicating    */
    /* the application is not running.                                    */
    /*                                                                    */

/*--------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


/*--------------------------------------------------------------------*/
    /*                                                                    */
    /* If the application has no actions to do on the Start action code   */
    /* and will become active as soon as the takeover IP address is       */
    /* activated, then this code should be uncommented.  This code will   */
    /* determine if all CRGs that this application CRG is dependent upon   */
    /* are ready.  If this check fails, return failure from the action     */
    /* code.                                                               */
    /*                                                                    */

```

```
/*------------------------------------------------------------------*/
/*      if (QcstSuccessful != checkDependCrgDataArea(MaxWaitSeconds))      */
/*         return QcstFailWithOutRestart;                                   */

  }
  else {

/*------------------------------------------------------------------*/
    /*                                                                    */
    /* This node is one of the other backup nodes or it is a replicate     */
    /* node.  If there is anything those nodes must do, do it here.  If */
    /* not, remove this else block.                                        */
    /*                                                                    */

/*------------------------------------------------------------------*/


/*------------------------------------------------------------------*/
    /*                                                                    */
    /* Ensure the data area status on this nodes starts out indicating     */
    /* the application is not running.                                     */
    /*                                                                    */

/*------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }

  return QcstSuccessful;
}  /* end switchPrimary()                                                 */


/***********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcAddNode                                     */
/*                                                                    */
/* The QcstAddNodeToRcvyDomain API was called.  A new node is being added  */
/* to the recovery domain of a cluster resource group.                 */
/*                                                                    */
/* Things to consider:                                                 */
/*   - A new node is being added to the recovery domain.  See the          */
/*     considerations in the createCrg() function.                     */
/*   - If this CRG is enabled to use the QcstDistributeInformation API,    */
/*     the user queue needed by that API could be created at this time.    */
/*                                                                    */
/***********************************************************************/
static int addNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*--------------------------------------------------------------------*/

  /*                                                                    */
  /* Determine if I am the node being added.                            */
  /*                                                                    */

/*--------------------------------------------------------------------*/

  if (0 == memcmp(&crgData->This_Nodes_ID,
                  &crgData->Changing_Node_ID,
                  sizeof(Qcst_Node_Id_t)))
  {

/*------------------------------------------------------------------*/

    /*                                                                    */
    /* Set the status of the data area on this new node.                 */
    /*                                                                   */

/*-----------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


/*-----------------------------------------------------------------------*/

    /*                                                                   */
    /* Create the queue needed by the Distribute Information API.        */
    /*                                                                   */

/*-----------------------------------------------------------------------*/

    if (0 == memcmp(&crgData->DI_Queue_Name,
                    Nulls,
                    sizeof(crgData->DI_Queue_Name)))
    {
    }
  }

  return QcstSuccessful;
}  /* end addNode()
      */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcRemoveNode                                       */
/*                                                                         */
/* The QcstRemoveNodeFromRcvyDomain or the QcstRemoveClusterNodeEntry      */
/* API was called.  A node is being removed from the recovery domain of    */
/* a cluster resource group or it is being removed entirely from the       */
/* cluster.                                                                */
/*                                                                         */
/* This action code is seen by:                                            */
/*  For the QcstRemoveClusterNodeEntry API:                                */
/*    - If the removed node is active and the CRG is Inactive, all nodes in*/
/*      the recovery domain including the node being removed see this     */
/*      action code.  The nodes NOT being removed see action code dependent*/
/*      data of QcstNodeFailure.                                           */
/*    - If the removed node is active and the CRG is Active, the node being*/
/*      removed sees the Remove Node action code.  All other nodes in the  */
/*      recovery domain see an action code of Failover and action code     */
/*      dependent data of QcstNodeFailure.                                 */
/*    - If the node being removed is not active in the cluster, all nodes  */
/*      in the recovery domain will see this action code.                  */
/*  For the QcstRemoveNodeFromRcvyDomain API:                              */
/*    - All nodes see the Remove Node action code regardless of whether or */
/*      not the CRG is Active.  Action code dependent data of              */
/*      QcstRmvRcvyDmnNode will also be passed.                            */
/*                                                                         */
/* Things to consider:                                                     */
/*   - You may want to cleanup the removed node by deleting objects no     */
/*     longer needed there.                                                */
/*   - The job running the application is canceled and the IP takeover    */
/*     address is ended after the exit program is called if this is the    */
/*     primary node and the CRG is active.                                 */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here or consolidate all logic  */
/*     to end the application in the cancelHandler() since it will be      */
/*     invoked for all Cluster Resource Services APIs which must end the   */
/*     application on the current primary.                                 */
/*                                                                         */
/***************************************************************************/
static int rmvNode(int role,
|                    int priorRole,
|                    Qcst_EXTP0100_t *crgData,
|                    EpData *epData) {
|
|
| /*---------------------------------------------------------------------*/
|
|   /*                                                                    */
|   /* Determine if I am the node being removed.                          */
|   /*                                                                    */
|
| /*---------------------------------------------------------------------*/
|
|   if (0 == memcmp(&crgData->This_Nodes_ID,
|                   &crgData->Changing_Node_ID,
|                   sizeof(Qcst_Node_Id_t)))
|   {
|
| /*-------------------------------------------------------------------*/
|       /*                                                                */
|       /* End the application if it is running on this node.            */
|       /*                                                                */
|
| /*-------------------------------------------------------------------*/
|       endApplication(QcstCrgAcRemoveNode, role, priorRole, crgData,
| epData);
|
|   }
|   return QcstSuccessful;
| }  /* end rmvNode                                                       */
|
|
| /***********************************************************************/
| /*                                                                     */
| /* Action code = QcstCrgAcChange                                       */
| /*                                                                     */
| /* The QcstChangeClusterResourceGroup API was called.  Some attribute  */
| /* or information stored in the cluster resource group object is being */
| /* changed.  Note that not all changes to the CRG object cause the exit */
| /* program to be called.  As of V5R1M0, only these changes will cause the */
| /* exit program to be called-                                          */
| /*   - the current recovery domain is being changed                    */
| /*   - the preferred recovery domain is being changed                  */
| /*                                                                     */
| /* If any of the above changes are being made but additionally the exit */
| /* program is being changed to *NONE, the exit program is not called.  */
| /*                                                                     */
| /* Things to consider:                                                 */
| /*   - None unless changing the recovery domain affects information or */
| /*     processes for this cluster resource group.  Note that the primary */
| /*     node cannot be changed with the QcstChangeClusterResourceGroup API */
| /*     if the CRG is active.                                           */
| /*                                                                     */
| /***********************************************************************/
| static int chgCrg(int role,
|                   int priorRole,
|                   Qcst_EXTP0100_t *crgData,
|                   EpData *epData) {
|
|   return QcstSuccessful;
| }  /* end chgCrg()                                                      */
|
|
| /***********************************************************************/
| /*                                                                     */
| /* Action code = QcstCrgAcDeleteCommand                                */
| /*                                                                     */
```

```
/* The Delete Cluster Resource Group (DLTCRG) CL command has been called   */
/* to delete a cluster resource group object, the QcstDeleteCluster API    */
/* has been called, or the QcstRemoveClusterNodeEntry API has been called. */
/* In each case, cluster resource services is not active on the cluster    */
/* node where the command or API was called.  Thus, this function is not   */
/* distributed cluster wide but occurs only on the node where the CL       */
/* command or API was called.                                              */
/*                                                                         */
/* If the QcstDeleteCluster API was used, action code dependent data of    */
/* QcstDltCluster is passed.                                               */
/*                                                                         */
/* See the considerations in the deleteCrg() function                      */
/*                                                                         */
/***************************************************************************/
static int deleteCrgWithCmd(int role,
                            int doesNotApply,
                            Qcst_EXTP0100_t *crgData,
                            EpData *epData) {

  return QcstSuccessful;
}  /* end deleteCrgWithCmd()                                               */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgEndNode                                            */
/*                                                                         */
/* The QcstEndClusterNode API was called or a CRG job was canceled.       */
/*                                                                         */
/* The QcstCrgEndNode action code is passed to the exit program only on the*/
/* node being ended or where the CRG job was canceled.  On the node where */
/* a Cluster Resource Services job is canceled, action code dependent data*/
/* of QcstMemberFailure will be passed.                                    */
/* When Cluster Resource Services ends on this node or the CRG job ends, it*/
/* will cause all other nodes in the cluster to go through failover        */
/* processing.  The action code passed to all other nodes will be          */
/* QcstCrgAcFailover.   Those nodes will see action code dependent data of */
/* QcstMemberFailure if a CRG job is canceled or QcstNodeFailure if the   */
/* node is ended.                                                          */
/*                                                                         */
/* Things to consider:                                                     */
/*   - The job running the application is canceled and the IP takeover    */
/*     address is ended after the exit program is called if this is the    */
/*     primary node and the CRG is active.                                 */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here.                          */
/*                                                                         */
/***************************************************************************/
static int endNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {



/*-----------------------------------------------------------------------*/
  /*                                                                       */
  /* End the application if it is running on this node.                   */
  /*                                                                       */

/*-----------------------------------------------------------------------*/
  endApplication(QcstCrgEndNode, role, priorRole, crgData, epData);

  return QcstSuccessful;
}  /* end endNode()                                                        */


/***************************************************************************/
```

```
/*                                                                         */
/* Action code = QcstCrgAcChgNodeStatus                                    */
/*                                                                         */
/* The QcstChangeClusterNodeEntry API was called.  The status of a node    */
/* is being changed to failed.  This API is used to inform cluster resource*/
/* services that the node did not partition but really failed.             */
/*                                                                         */
/* Things to consider:                                                     */
/*   - The exit program was called previously with an action code of       */
/*     QcstCrgAcEnd if the CRG was active or an action code of             */
/*     QcstCrgAcFailover if the CRG was inactive because cluster resource  */
/*     services thought the cluster had become partitioned.  The user is   */
/*     now telling cluster resource services that the node really failed   */
/*     instead of partitioned.  The exit program has something to do only  */
/*     if it performed some action previously that needs to be changed now */
/*     that node failure can be confirmed.                                 */
/*                                                                         */
/***************************************************************************/
static int chgNodeStatus(int role,
                         int priorRole,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {

  return QcstSuccessful;
}  /* end chgNodeStatus()                                                  */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcCancelFailover                                   */
/*                                                                         */
/* Cluster resource services on the primary node has failed or ended       */
/* for this cluster resource group.  A message was sent to the failover    */
/* message queue specified for the CRG, and the result of that message     */
/* was to cancel the failover.  This will change the status of the CRG to  */
/* inactive and leave the primary node as primary.                         */
/*                                                                         */
/* Things to consider:                                                     */
/*   - The primary node is no longer participating in cluster activities.  */
/*     The problem which caused the primary node to fail should be fixed   */
/*     so that the CRG may be started again.                               */
/*                                                                         */
/***************************************************************************/
static int cancelFailover(int role,
                          int priorRole,
                          Qcst_EXTP0100_t *crgData,
                          EpData *epData) {

  return QcstSuccessful;
}  /* end cancelFailover()                                                 */


/***************************************************************************/
/*                                                                         */
/* Action code = exit program does not know it yet                         */
/*                                                                         */
/* A new action code has been passed to this exit program.  This can occur */
/* after a new i5/OS release has been installed and some new cluster API  */
/* was called or some new cluster event occurred.  The logic in this exit  */
/* program has not yet been updated to understand the new action code.    */
/*                                                                         */
/* Two different strategies could be used for the new action code.  The    */
/* correct strategy is dependent upon the kinds of things this particular  */
/* exit program does for the application.                                  */
/*                                                                         */
/* One strategy is to not do anything and return a successful return code. */
/* This allows the new cluster API or event to run to completion.  It      */
/* allows the function to be performed even though this exit program       */
/* did not understand the new action code.  The risk, though, is that the  */
/* exit program should have done something and it did not.  At a minimum,  */
/* you may want to log some kind of error message about what happened so   */
/* that programming can investigate and get the exit program updated.      */
/*                                                                         */
/* The opposite strategy is to return an error return code such as         */
/* QcstFailWithRestart.  Of course doing this means that the new cluster   */
/* API or event cannot be used until the exit program is updated for the   */
/* new action code.  Again, logging some kind of error message for         */
/* programming to investigate would be worthwhile.                         */
/*                                                                         */
/* Only the designer of the exit program can really decide which is the    */
/* better course of action.                                                */
/*                                                                         */
/***************************************************************************/
static int newActionCode(int role,
                         int doesNotApply,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {


/*-----------------------------------------------------------------------*/
  /*                                                                       */
  /* Add logic to log an error somewhere - operator message queue, job     */
  /* log, application specific error log, etc. so that the exit program    */
  /* gets updated to properly handle the new action code.                  */
  /*                                                                       */
  /* Note that if this is left coded as it is, this is the "don't do       */
  /* anything" strategy described in the prologue above.                   */
  /*                                                                       */

/*-----------------------------------------------------------------------*/

  return QcstSuccessful;
}  /* end newActionCode()                                                  */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Note: The exit program is never called with an undo action code for    */
/* any of these prior action codes:                                        */
/*   QcstCrgAcChgNodeStatus                                                */
/*   QcstCrgAcDelete                                                       */
/*   QcstCrgAcDeleteCommand                                                */
/*   QcstCrgEndNode                                                        */
/*   QstCrgAcRemoveNode (If the node being removed is active in the        */
/*                       cluster and the API is Remove Cluster Node.       */
/*                       The Remove Node From Recovery Domain will call    */
/*                       with Undo and the Remove Cluster Node API will    */
/*                       call with Undo if the node being removed is       */
/*                       inactive.                                         */
/*   QcstCrgAcRestart                                                      */
/*   QcstCrgAcUndo                                                         */
/*                                                                         */
/* APIs that call an exit program do things in 3 steps.                    */
/*   1. Logic which must be done prior to calling the exit program.        */
/*   2. Call the exit program.                                             */
/*   3. Logic which must be done after calling the exit program.           */
/*                                                                         */
/* Any errors that occur during steps 2 or 3 result in the exit program    */
/* being called again with the undo action code.  This gives the exit      */
/* program an opportunity to back out any work performed when it was first */
/* called by the API.  The API will also be backing out any work it        */
/* performed trying to return the state of the cluster and cluster objects */
/* to what it was before the API was called.                            */
/*                                                                      */
/* It is suggested that the following return codes be returned for the  */
/* specified action code as that return code will result in the most    */
/* appropriate action being taken.                                      */
/*                                                                      */
/*   QcstCrgAcInitialize: QcstSuccessful; The CRG is not created.       */
/*   QcstCrgAcStart:      QcstSuccessful; The CRG is not started.       */
/*   QcstCrgAcEnd:        QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.                 */
/*   QcstCrgAcReJoin:     QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.                 */
/*   QcstCrgAcFailover:   QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.                 */
/*   QcstCrgAcSwitchover: QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.                 */
/*   QcstCrgAcAddNode:    QcstSuccessful; The node is not added.        */
/*   QcstCrgAcRemoveNode: QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.                 */
/*   QcstCrgAcChange:     QcstSuccessful; The recovery domain is not    */
/*                                        changed.                      */
/*                                                                      */
/************************************************************************/
static int undoPriorAction(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*---------------------------------------------------------------------*/
  /*                                                                    */
  /* The prior action code defines what the exit program was doing when */
  /* it failed, was canceled, or returned a non successful return code.  */
  /*                                                                    */

/*---------------------------------------------------------------------*/
  if (crgData->Prior_Action_Code &lt;= MaxAc )
    return (*undoFcn[crgData-&lt;Prior_Action_Code])
                                          (role, priorRole, crgData,
epData);
  else

/*-------------------------------------------------------------------*/
    /*                                                                  */
    /* IBM has defined a new action code in a new operating system release */
    /* and this exit program has not yet been updated to handle it.  Take a*/
    /* default action for now.                                          */
    /*                                                                  */

/*-------------------------------------------------------------------*/
    return newActionCode(role, priorRole, crgData, epData);
}  /* end undoPriorAction()                                              */


/************************************************************************/
/*                                                                      */
/* Action code = QcstCrgAcUndo                                          */
/*                                                                      */
/* Prior action code = QcstCrgAcInitialize                              */
/*                                                                      */
/* Things to consider:                                                  */
/*   The CRG will not be created.  Objects that might have been created */
/*   on nodes in the recovery domain should be deleted since a subsequent  */
/*   create could fail if those objects already exist.                     */
/*                                                                         */
/***************************************************************************/
static int undoCreateCrg(int role,
                         int doesNotApply,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {

  return QcstSuccessful;
}  /* end undoCreateCrg()                                                 */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Prior action code = QcstCrgAcStart                                      */
/*                                                                         */
/* Things to consider:                                                     */
/*   Cluster Resource Services failed when it was finishing the Start CRG  */
/*   API after it had already called the exit program with the Start       */
/*   Action code.                                                          */
/*                                                                         */
/*   On the primary node, the exit program job which is running the        */
/*   application will be canceled.  The exit program will then be called  */
/*   with the Undo action code.                                            */
/*                                                                         */
/*   All other nodes in the recovery domain will be called with the Undo   */
/*   action code.                                                          */
/*                                                                         */
/***************************************************************************/
static int undoStartCrg(int role,
                        int doesNotApply,
                        Qcst_EXTP0100_t *crgData,
                        EpData *epData) {

  return QcstSuccessful;
}  /* end undoStartCrg()                                                   */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Prior action code = QcstCrgAcEnd                                        */
/*                                                                         */
/* Things to consider:                                                     */
/*   The CRG will not be ended.  If the exit program did anything to bring */
/*   down the application it can either restart the application or it can  */
/*   decide to not restart the application.  If the application is not     */
/*   restarted, the return code should be set to QcstFailWithOutRestart so */
/*   the status of the CRG is set to Indoubt.                              */
/*                                                                         */
/***************************************************************************/
static int undoEndCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoEndCrg()                                                     */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                        */
/* Prior action code = QcstCrgAcReJoin                                    */
/*                                                                        */
/* Things to consider:                                                    */
/*   An error occurred which won't allow the member to join this CRG      */
/*   group.  Anything done for the Join action code needs to be looked at */
/*   to see if something must be undone if this member is not an active   */
/*   member of the CRG group.                                             */
/*                                                                        */
/**************************************************************************/
static int undoMemberIsJoining(int role,
                               int doesNotApply,
                               Qcst_EXTP0100_t *crgData,
                               EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoMemberIsJoining()                                           */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcUndo                                            */
/*                                                                        */
/* Prior action code = QcstCrgAcFailover                                  */
/*                                                                        */
/* Things to consider:                                                    */
/*   This does not mean that the node failure or failing member is being  */
/*   undone.  That failure is irreversible.  What it does mean is that the */
/*   exit program returned an error from the Failover action code or      */
/*   Cluster Resource Services ran into a problem after it called the exit */
/*   program.  If the CRG was active when Failover was attempted, it is   */
/*   not at this point.  End the resilient resource and expect a human to  */
/*   look into the failure.  After the failure is corrected, the CRG will  */
/*   must be started with the Start CRG API.                           */
/*                                                                        */
/*                                                                        */
/**************************************************************************/
static int undoMemberIsLeaving(int role,
                               int doesNotApply,
                               Qcst_EXTP0100_t *crgData,
                               EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoMemberIsLeaving()                                           */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcUndo                                            */
/*                                                                        */
/* Prior action code = QcstCrgAcSwitchover                                */
/*                                                                        */
/* Things to consider:                                                    */
/*   Some error occurred after the point of access was moved from the     */
/*   original primary and before it could be brought up on the new primary.*/
/*   The IP address was ended on the original primary before moving the   */
/*   point of access but is started on the original primary again.  Cluster*/
/*   Resource Services will now attempt to move the point of access back  */
/*   to the original primary.  The application exit program and IP takeover*/
/*   address will be started on the original primary.                     */
/*                                                                        */
/*                                                                        */
/**************************************************************************/
static int undoSwitchPrimary(int role,
                             int doesNotApply,
                             Qcst_EXTP0100_t *crgData,
                             EpData *epData) {
```

```

  return QcstFailWithOutRestart;
}  /* end undoSwitchPrimary()                                            */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Prior action code = QcstCrgAcAddNode                                    */
/*                                                                         */
/* Things to consider:                                                     */
/*   If objects were created on the new node, they should be removed so    */
/*   that a subsequent Add Node to aRecovery Domain does not fail if it    */
/*   attempts to create objects again.                                     */
/*                                                                         */
/*                                                                         */
/***************************************************************************/
static int undoAddNode(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  return QcstSuccessful;
}  /* end undoAddNode()                                                  */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Prior action code = QcstCrgAcRemoveNode                                 */
/*                                                                         */
/* Things to consider:                                                     */
/*   The node is still in the recovery domain.  If objects were removed    */
/*   from the node, they should be added back.                             */
/*                                                                         */
/***************************************************************************/
static int undoRmvNode(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoRmvNode()                                                  */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Prior action code = QcstCrgAcChange                                     */
/*                                                                         */
/* Things to consider:                                                     */
/*   Changes to the CRG will be backed out so that the CRG and its         */
/*   recovery domain look just like it did prior to the attempted change.  */
/*   Any changes the exit program made should also be backed out.          */
/*                                                                         */
/***************************************************************************/
static int undoChgCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {

  return QcstSuccessful;
}  /* end undoChgCrg()                                                   */

```

```
/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcUndo                                             */
/*                                                                         */
/* Prior action code = QcstCrgAcCancelFailover                             */
/*                                                                         */
/* Things to consider:                                                     */
/*   This does not mean that the node failure or failing member is being   */
/*   undone.  That failure is irreversible.  What it does mean is that     */
/*   Cluster Resource Services ran into a problem after it called the exit */
/*   program.  The CRG will be InDoubt regardless of what is returned from */
/*   this exit program call.  Someone will need to manually look into the  */
/*   the failure.  After the failure is corrected, the CRG will must be */
/*   started with the Start CRG API.                                       */
/*                                                                         */
/*                                                                         */
/***************************************************************************/
static int undoCancelFailover(int role,
                              int doesNotApply,
                              Qcst_EXTP0100_t *crgData,
                              EpData *epData) {

  return QcstSuccessful;
}  /* end undoCancelFailover()                                             */


/***************************************************************************/
/*                                                                         */
/* A simple routine to take a null terminated object name and a null       */
/* terminated library name and build a 20 character non-null terminated    */
/* qualified name.                                                         */
/*                                                                         */
/***************************************************************************/
static void bldDataAreaName(char *objName, char* libName, char *qualName) {

  memset(qualName, 0x40, 20);
  memcpy(qualName, objName, strlen(objName));
  qualName += 10;
  memcpy(qualName, libName, strlen(libName));
  return;
}  /* end bldDataAreaName                                                  */


/***************************************************************************/
/*                                                                         */
/* The data area is checked to see if all the CRGs that this application   */
/* is dependent upon are ready.  If they are not ready, a wait for a       */
/* certain amount of time is performed and the data area is checked again. */
/* This check, wait loop continues until all dependent CRGs become ready or*/
/* until the maximum wait time has been reached.                           */
/* The length of the wait can be changed to some other value if a          */
/* particular situation would be better with shorter or longer wait times. */
/*                                                                         */
/*                                                                         */
/***************************************************************************/
static int checkDependCrgDataArea(unsigned int maxWaitTime) {

  Qus_EC_t errCode = { sizeof(Qus_EC_t), 0 };
  char dataAreaName[20];
  struct {
    Qwc_Rdtaa_Data_Returned_t stuff;
    char ready;
  } data;


/*---------------------------------------------------------------------*/
```

```
    /*                                                                    */
    /* This is an accumulation of the time waited for the dependent CRGs to  */
    /* become ready.                                                      */
    /*                                                                    */

  /*----------------------------------------------------------------------*/
    unsigned int timeWaited = 0;


  /*----------------------------------------------------------------------*/
    /*                                                                    */
    /* Build definition of the amount of time to wait.                    */
    /*                                                                    */

  /*----------------------------------------------------------------------*/
    _MI_Time   timeToWait;
    int hours     = 0;
    int minutes   = 0;
    int seconds   = WaitSecondsIncrement;
    int hundreths = 0;
    short int options = _WAIT_NORMAL;
    mitime( &timeToWait, hours, minutes, seconds, hundreths );


  /*----------------------------------------------------------------------*/
    /*                                                                    */
    /* Build the qualified name of the data area.                         */
    /*                                                                    */

  /*----------------------------------------------------------------------*/
    bldDataAreaName(DependCrgDataArea, ApplLib, dataAreaName);


  /*----------------------------------------------------------------------*/
    /*                                                                    */
    /* Get the data from the data area that indicates whether or not the  */
    /* CRGs are all ready.  This data area is updated by the High         */
    /* Availability Business Partners when it is ok for the application to */
    /* proceed.                                                           */
    /*                                                                    */

  /*----------------------------------------------------------------------*/
    QWCRDTAA(&data,
             sizeof(data),
             dataAreaName,
             offsetof(Qcst_HAAPPO_t,Data_Status)+1,  /* API wants a 1 origin */
             sizeof(data.ready),
             &errCode);


  /*----------------------------------------------------------------------*/
    /*                                                                    */
    /* If the dependent CRGs are not ready, wait for a bit and check again. */
    /*                                                                    */

  /*----------------------------------------------------------------------*/
    while (data.ready != Data_Available) {


  /*--------------------------------------------------------------------*/
      /*                                                                  */
      /* If the dependent CRGs have not become ready by the time we have  */
      /* waited our maximum wait time, return an error.  Consider logging */
      /* some message to describe why the application did not start so that */
      /* the problem can be looked into.                                  */
      /*                                                                  */

```

```
/*------------------------------------------------------------------*/
    if (timeWaited >= maxWaitTime)
      return QcstFailWithOutRestart;


/*------------------------------------------------------------------*/
    /*                                                                */
    /* Wait to allow the data CRGs to become ready.                   */
    /*                                                                */

/*------------------------------------------------------------------*/
    waittime(&timeToWait, options);
    timeWaited += WaitSecondsIncrement;


/*------------------------------------------------------------------*/
    /*                                                                */
    /* Get information from the data area again to see if the data CRGs are*/
    /* ready.                                                         */
    /*                                                                */

/*------------------------------------------------------------------*/
    QWCRDTAA(&data,
             sizeof(data),
             dataAreaName,
             offsetof(Qcst_HAAPPO_t,Data_Status)+1,  /* API wants a 1 origin */
             sizeof(data.ready),
             &errCode);
  }

  return QcstSuccessful;
}  /* end checkDependCrgDataArea                                        */


/**********************************************************************/
/*                                                                    */
/* The application CRG data area is updated to indicate that the      */
/* application is running or to indicate it is not running.  This data area*/
/* information is used by the High Availability Business Partners to  */
/* coordinate the switchover activities between CRGs that have dependencies*/
/* on each other.                                                     */
/*                                                                    */
/**********************************************************************/
static void setApplCrgDataArea(char status) {

  char cmd[54];
  char cmdEnd[3] = {0x00, ')', 0x00};


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set up the CL command string with the data area library name, the data*/
  /* area name, and the character to put into the data area.  Then run the */
  /* CL command.                                                      */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  memcpy(cmd, "CHGDTAARA DTAARA(", strlen("CHGDTAARA DTAARA(")+1);
  strcat(cmd, ApplLib);
  strcat(cmd, "/");
  strcat(cmd, ApplCrgDataArea);
  strcat(cmd, " (425 1)) VALUE(");                            /* @A1C */
  cmdEnd[0] = status;
  strcat(cmd, cmdEnd);

  system(cmd);

  return;
}  /* end setApplCrgDataArea                                          */


/***************************************************************************/
/*                                                                         */
/* This function is called any time the exit program receives an exception */
/* not specifically monitored for by some other exception handler.  Add    */
/* appropriate logic to perform cleanup functions that may be required.    */
/* A failure return code is then set and control returns to the operating  */
/* system.  The job this exit program is running in will then end.         */
/*                                                                         */
/* When this function gets called, myData->role may still contain the      */
/* UnknownRole value if an exception occurred before this node's role      */
/* value was set.  To be completely correct, the role should be tested     */
/* for UnknownRole before making any decisions based upon the value of     */
/* role.                                                                   */
/*                                                                         */
/***************************************************************************/
static void unexpectedExceptionHandler(_INTRPT_Hndlr_Parms_T
*exData) {


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Get a pointer to the structure containing data we passed to the       */
  /* exception handler.                                                    */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  HandlerDataT *myData = (HandlerDataT *)exData->Com_Area;


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Perform as much cleanup function as necessary.  Some global state     */
  /* information may must be kept so the exception handler knows what   */
  /* steps were completed before the failure occurred and thus knows what  */
  /* cleanup steps must be performed.  This state information could be     */
  /* kept in the HandlerDataT structure or it could be kept in some other  */
  /* location that this function can address.                              */
  /*                                                                       */

/*-------------------------------------------------------------------------*/



/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* If this is the primary node and the application was started, end it.  */
  /* The application is ended because the exit program will be called again*/
  /* with the Restart action code and we want the restartCrg() function to */
  /* always work the same way.  In addition, ending the application may    */
  /* clear up the condition that caused the exception that got us here.    */
  /* If possible, warn users and have them stop using the application so   */
  /* things are done in an orderly manner.                                 */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  endApplication(myData->actionCode,
                 myData->role,
                 myData->priorRole,
                 myData->crgData,
                 myData->epData);


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Set the exit program return code.                                  */
  /*                                                                    */

/*--------------------------------------------------------------------*/
  *myData->retCode = QcstFailWithRestart;


/*--------------------------------------------------------------------*/
  /*                                                                    */
  /* Let the exception percolate up the call stack.                 */
  /*                                                                    */

/*--------------------------------------------------------------------*/
  return;
}  /* end unexpectedExceptionHandler                                    */


/***********************************************************************/
/*                                                                      */
/* This function is called any time the job this exit program is running in*/
/* is canceled.  The job could be canceled due to any of the following   */
/* (the list is not intended to be all inclusive)-                      */
/*   - an API cancels an active application CRG.  The End CRG, Initiate    */
/*     Switchover, End Cluster Node, Remove Cluster Node or Delete Cluster */
/*     API cancels the job which was submitted when the exit program was   */
/*     called with a Start action code.                                 */
/*   - operator cancels the job from some operating system display such as */
/*     Work with Active Jobs                                            */
/*   - the subsystem this job is running in is ended                    */
/*   - all subsystems are ended                                         */
/*   - the system is powered down                                       */
/*   - an operating system machine check occurred                       */
/*                                                                      */
/* When this function gets called, myData->role may still contain the     */
/* UnknownRole value if cancelling occurred before this node's role       */
/* value was set.  To be completely correct, the role should be tested    */
/* for UnknownRole before making any decisions based upon the value of    */
/* role.                                                                */
/*                                                                      */
/***********************************************************************/
static void cancelHandler(_CNL_Hndlr_Parms_T *cnlData) {


/*--------------------------------------------------------------------*/
  /*                                                                    */
  /* Get a pointer to the structure containing data we passed to the      */
  /* cancel handler.                                                    */
  /*                                                                    */

/*--------------------------------------------------------------------*/
  HandlerDataT *myData = (HandlerDataT *)cnlData->Com_Area;


/*--------------------------------------------------------------------*/
  /*                                                                    */
  /* Perform as much cleanup function as necessary.  Some global state     */
  /* information may must be kept so the cancel handler knows what      */
  /* steps were completed before the job was canceled and thus knows if   */
  /* the function had really completed successfully or was only partially  */
  /* complete and thus needs some cleanup to be done.  This state          */
  /* information could be kept in the HandlerDataT structure or it could   */
  /* be kept in some other location that this function can address.       */
  /*                                                                    */

/*--------------------------------------------------------------------*/
```

```
/*------------------------------------------------------------------*/

  /*                                                                  */
  /* This job is being canceled.  If I was running the application as a   */
  /* result of the Start or Restart action codes, end the application now. */
  /* This job is being canceled because a Switch Over or some other       */
  /* Cluster Resource Services API was used which affects the primary node */
  /* or someone did a cancel job with a CL command, from a system display, */
  /* etc.                                                                  */


/*------------------------------------------------------------------*/
  endApplication(myData->actionCode,
                 myData->role,
                 myData->priorRole,
                 myData->crgData,
                 myData->epData);


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* Set the exit program return code.                                */
  /*                                                                  */

/*------------------------------------------------------------------*/
  *myData->retCode = QcstSuccessful;


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* Return to the operating system for final ending of the job.      */
  /*                                                                  */

/*------------------------------------------------------------------*/
  return;
}  /* end cancelHandler                                               */


/**********************************************************************/
/*                                                                    */
/* A common routine used to end the application by various action code */
/* functions, the exception handler, and the cancel handler.          */
/*                                                                    */
/**********************************************************************/
static void endApplication(unsigned int actionCode,
                           int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {

  if (  role == QcstPrimaryNodeRole
      &&
        crgData->Original_Cluster_Res_Grp_Stat == QcstCrgActive)
{

/*----------------------------------------------------------------*/
    /*                                                                */
    /* Add logic to end the application here.  You may need to add logic */
    /* to determine if the application is still running because this   */
    /* function could be called once for an action code and again from */
    /* the cancel handler (End CRG is an example).                     */
    /*                                                                */

/*----------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
    /*                                                                 */
    /* After the application has ended, update the data area to indicate   */
    /* the application is no longer running.                            */
    /*                                                                 */

/*-------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }

  return;
}  /* end endApplication                                               */


/***************************************************************************/
/*                                                                         */
/* Print out the data passed to this program.                              */
/*                                                                         */
/***************************************************************************/
static void printParms(int actionCode,
                       int role,
                       int priorRole,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  unsigned int i;
  char *str;

  /* Print the action code.                                              */
  printf("%s", "Action_Code = ");
  printActionCode(actionCode);

  /* Print the action code dependent data.                               */
  printf("%s", "    Action_Code_Dependent_Data = ");
  switch (crgData->Action_Code_Dependent_Data) {
    case QcstNoDependentData:  str = "QcstNoDependentData";
                               break;
    case QcstMerge:            str = "QcstMerge";
                               break;
    case QcstJoin:             str = "QcstJoin";
                               break;
    case QcstPartitionFailure: str = "QcstPartitionFailure";
                               break;
    case QcstNodeFailure:      str = "QcstNodeFailure";
                               break;
    case QcstMemberFailure:    str = "QcstMemberFailure";
                               break;
    case QcstEndNode:          str = "QcstEndNode";
                               break;
    case QcstRemoveNode:       str = "QcstRemoveNode";
                               break;
    case QcstApplFailure:      str = "QcstApplFailure";
                               break;
    case QcstResourceEnd:      str = "QcstResourceEnd";
                               break;
    case QcstDltCluster:       str = "QcstDltCluster";
                               break;
    case QcstRmvRcvyDmnNode:   str = "QcstRmvRcvyDmnNode";
                               break;
    case QcstDltCrg:           str = "QcstDltCrg";
                               break;
    default: str = "unknown action code dependent data";
  }
  printf("%s \n", str);
```

```
|   /* Print the prior action code.                                        */
|   printf("%s", "  Prior_Action_Code = ");
|   if (crgData->Prior_Action_Code)
|     printActionCode(crgData->Prior_Action_Code);
|   printf("\n");
|
|   /* Print the cluster name.                                             */
|   printStr("  Cluster_Name = ",
|            crgData->Cluster_Name, sizeof(Qcst_Cluster_Name_t));
|
|   /* Print the CRG name.                                                 */
|   printStr("  Cluster_Resource_Group_Name = ",
|            crgData->Cluster_Resource_Group_Name,
| sizeof(Qcst_Crg_Name_t));
|
|   /* Print the CRG type.                                                 */
|   printf("%s \n", "  Cluster_Resource_Group_Type =
| QcstCrgApplResiliency");
|
|   /* Print the CRG status.                                               */
|   printf("%s", "  Cluster_Resource_Group_Status = ");
|   printCrgStatus(crgData->Cluster_Resource_Group_Status);
|
|   /* Print the CRG original status.                                      */
|   printf("%s", "  Original_Cluster_Res_Grp_Stat = ");
|   printCrgStatus(crgData->Original_Cluster_Res_Grp_Stat);
|
|   /* Print the Distribute Information queue name.                        */
|   printStr("  DI_Queue_Name = ",
|            crgData->DI_Queue_Name,
| sizeof(crgData->DI_Queue_Name));
|   printStr("  DI_Queue_Library_Name = ",
|            crgData->DI_Queue_Library_Name,
|            sizeof(crgData->DI_Queue_Library_Name));
|
|   /* Print the CRG attributes.                                           */
|   printf("%s", "  Cluster_Resource_Group_Attr = ");
|   if (crgData->Cluster_Resource_Group_Attr &
| QcstTcpConfigByUsr)
|     printf("%s", "User Configures IP Takeover Address");
|   printf("\n");
|
|   /* Print the ID of this node.                                          */
|   printStr("  This_Nodes_ID = ",
|            crgData->This_Nodes_ID, sizeof(Qcst_Node_Id_t));
|
|   /* Print the role of this node.                                        */
|   printf("%s %d \n", "  this node's role = ", role);
|
|   /* Print the prior role of this node.                                  */
|   printf("%s %d \n", "  this node's prior role = ", priorRole);
|
|   /* Print which recovery domain this role comes from.                   */
|   printf("%s", "  Node_Role_Type = ");
|   if (crgData->Node_Role_Type == QcstCurrentRcvyDmn)
|     printf("%s \n", "QcstCurrentRcvyDmn");
|   else
|     printf("%s \n", "QcstPreferredRcvyDmn");
|
|   /* Print the ID of the changing node (if any).                         */
|   printStr("  Changing_Node_ID = ",
|            crgData->Changing_Node_ID, sizeof(Qcst_Node_Id_t));
|
|   /* Print the role of the changing node (if any).                       */
|   printf("%s", "  Changing_Node_Role = ");
|   if (crgData->Changing_Node_Role == -3)
|     printf("%s \n", "*LIST");
```

```
  else if (crgData->Changing_Node_Role == -2)
   printf("%s \n", "does not apply");
  else
    printf("%d \n", crgData->Changing_Node_Role);

  /* Print the takeover IP address.                                        */
  printStr("  Takeover_IP_Address = ",
           crgData->Takeover_IP_Address,
sizeof(Qcst_TakeOver_IP_Address_t));

  /* Print the job name.                                                   */
  printStr("  Job_Name = ", crgData->Job_Name, 10);

  /* Print the CRG changes.                                                */
  printf("%s \n", "  Cluster_Resource_Group_Changes = ");
  if (crgData->Cluster_Resource_Group_Changes &
QcstRcvyDomainChange)
    printf("     %s \n", "Recovery domain changed");
  if (crgData->Cluster_Resource_Group_Changes &
QcstTakeOverIpAddrChange)
    printf("     %s \n", "Takeover IP address changed");

  /* Print the failover wait time.                                         */
  printf("%s", "Failover_Wait_Time = ");
  if (crgData->Failover_Wait_Time == QcstFailoverWaitForever)
     printf("%d %s \n", crgData->Failover_Wait_Time, "Wait
forever");
  else if (crgData->Failover_Wait_Time == QcstFailoverNoWait)
     printf("%d %s \n", crgData->Failover_Wait_Time, "No wait");
  else
     printf("%d %s \n", crgData->Failover_Wait_Time, "minutes");

  /* Print the failover default action.                                    */
  printf("%s", "Failover_Default_Action = ");
  if (crgData->Failover_Default_Action == QcstFailoverProceed)
     printf("%d %s \n", crgData->Failover_Default_Action,
"Proceed");
  else
     printf("%d %s \n", crgData->Failover_Default_Action,
"Cancel");

  /* Print the failover message queue name.                                */
  printStr("  Failover_Msg_Queue = ",
           crgData->Failover_Msg_Queue,
sizeof(crgData->Failover_Msg_Queue));
  printStr("  Failover_Msg_Queue_Lib = ",
           crgData->Failover_Msg_Queue_Lib,
           sizeof(crgData->Failover_Msg_Queue_Lib));

  /* Print the cluster version.                                            */
  printf("%s %d \n",
         "  Cluster_Version = ", crgData->Cluster_Version);

  /* Print the cluster version mod level                                   */
  printf("%s %d \n",
         "  Cluster_Version_Mod_Level = ",
         crgData->Cluster_Version_Mod_Level);

  /* Print the requesting user profile.                                    */
  printStr("  Req_User_Profile = ",
           crgData->Req_User_Profile,
sizeof(crgData->Req_User_Profile));

  /* Print the length of the data in the structure.                        */
  printf("%s %d \n",
         "  Length_Info_Returned = ",
crgData->Length_Info_Returned);
```

```
  /* Print the offset to the recovery domain array.                     */
  printf("%s %d \n",
         "  Offset_Rcvy_Domain_Array = ",
crgData->Offset_Rcvy_Domain_Array);

  /* Print the number of nodes in the recovery domain array.            */
  printf("%s %d \n",
         "  Number_Nodes_Rcvy_Domain = ",
crgData->Number_Nodes_Rcvy_Domain);

  /* Print the current/new recovery domain.                             */
  printRcvyDomain("   The recovery domain:",
                  crgData->Number_Nodes_Rcvy_Domain,
                  (Qcst_Rcvy_Domain_Array1_t *)
                  ((char *)crgData +
crgData->Offset_Rcvy_Domain_Array));

  /* Print the offset to the prior recovery domain array.               */
  printf("%s %d \n",
         "  Offset_Prior_Rcvy_Domain_Array = ",
         crgData->Offset_Prior_Rcvy_Domain_Array);

  /* Print the number of nodes in the prior recovery domain array.      */
  printf("%s %d \n",
         "  Number_Nodes_Prior_Rcvy_Domain = ",
         crgData->Number_Nodes_Prior_Rcvy_Domain);

  /* Print the prior recovery domain if one was passed.                 */
  if (crgData->Offset_Prior_Rcvy_Domain_Array) {
    printRcvyDomain("   The prior recovery domain:",
                    crgData->Number_Nodes_Prior_Rcvy_Domain,
                    (Qcst_Rcvy_Domain_Array1_t *)
                ((char *)crgData +
crgData->Offset_Prior_Rcvy_Domain_Array));
  }

  return;
}  /* end printParms                                                    */


/***************************************************************************/
/*                                                                       */
/* Print a string for the action code.                                   */
/*                                                                       */
/***************************************************************************/
static void printActionCode(unsigned int ac) {

  char *code;
  switch (ac) {
    case QcstCrgAcInitialize: code = "QcstCrgAcInitialize";
                              break;
    case QcstCrgAcStart:      code = "QcstCrgAcStart";
                              break;
    case QcstCrgAcRestart:    code = "QcstCrgAcRestart";
                              break;
    case QcstCrgAcEnd:        code = "QcstCrgAcEnd";
                              break;
    case QcstCrgAcDelete:     code = "QcstCrgAcDelete";
                              break;
    case QcstCrgAcReJoin:     code = "QcstCrgAcReJoin";
                              break;
    case QcstCrgAcFailover:   code = "QcstCrgAcFailover";
                              break;
    case QcstCrgAcSwitchover: code = "QcstCrgAcSwitchover";
                              break;
    case QcstCrgAcAddNode:    code = "QcstCrgAcAddNode";
```

```
                              break;
    case QcstCrgAcRemoveNode: code = "QcstCrgAcRemoveNode";
                              break;
    case QcstCrgAcChange:     code = "QcstCrgAcChange";
                              break;
    case QcstCrgAcDeleteCommand: code = "QcstCrgAcDeleteCommand";
                              break;
    case QcstCrgAcUndo:       code = "QcstCrgAcUndo";
                              break;
    case QcstCrgEndNode:      code = "QcstCrgEndNode";
                              break;
    case QcstCrgAcAddDevEnt:  code = "QcstCrgAcAddDevEnt";
                              break;
    case QcstCrgAcRmvDevEnt:  code = "QcstCrgAcRmvDevEnt";
                              break;
    case QcstCrgAcChgDevEnt:  code = "QcstCrgAcChgDevEnt";
                              break;
    case QcstCrgAcChgNodeStatus: code = "QcstCrgAcChgNodeStatus";
                              break;
    case QcstCrgAcCancelFailover: code = "QcstCrgAcCancelFailover";
                              break;
    case QcstCrgAcVerificationPhase: code =
"QcstCrgAcVerificationPhase";
                              break;
    default:                  code = "unknown action code";
                              break;
  }
  printf("%s", code);

  return;
}  /* end printActionCode                                                   */


/****************************************************************************/
/*                                                                          */
/* Print the CRG status.                                                    */
/*                                                                          */
/****************************************************************************/
static void printCrgStatus(int status) {

  char * str;
  switch (status) {
    case QcstCrgActive:               str = "QcstCrgActive";
                                      break;
    case QcstCrgInactive:             str= "QcstCrgInactive";
                                      break;
    case QcstCrgIndoubt:              str = "QcstCrgIndoubt";
                                      break;
    case QcstCrgRestored:             str = "QcstCrgRestored";
                                      break;
    case QcstCrgAddnodePending:       str =
"QcstCrgAddnodePending";
                                      break;
    case QcstCrgDeletePending:        str = "QcstCrgDeletePending";
                                      break;
    case QcstCrgChangePending:        str = "QcstCrgChangePending";
                                      break;
    case QcstCrgEndCrgPending:        str = "QcstCrgEndCrgPending";
                                      break;
    case QcstCrgInitializePending:    str =
"QcstCrgInitializePending";
                                      break;
    case QcstCrgRemovenodePending:    str =
"QcstCrgRemovenodePending";
                                      break;
    case QcstCrgStartCrgPending:      str =
"QcstCrgStartCrgPending";
```

```
                                      break;
    case QcstCrgSwitchOverPending:    str =
"QcstCrgSwitchOverPending";
                                      break;
    case QcstCrgDeleteCmdPending:     str =
"QcstCrgDeleteCmdPending";
                                      break;
    case QcstCrgAddDevEntPending:     str =
"QcstCrgAddDevEntPending";
                                      break;
    case QcstCrgRmvDevEntPending:     str =
"QcstCrgRmvDevEntPending";
                                      break;
    case QcstCrgChgDevEntPending:     str =
"QcstCrgChgDevEntPending";
                                      break;
    case QcstCrgChgNodeStatusPending: str =
"QcstCrgChgNodeStatusPending";
                                      break;
    default: str = "unknown CRG status";
  }
  printf("%s \n", str);

  return;
}  /* end printCrgStatus                                                    */


/**************************************************************************/
/*                                                                        */
/* Print the recovery domain.                                             */
/*                                                                        */
/**************************************************************************/
static void printRcvyDomain(char *str,
                            unsigned int count,
                            Qcst_Rcvy_Domain_Array1_t *rd) {

  unsigned int i;
  printf("\n %s \n", str);
  for (i=1; i&lt;=count; i++) {
    printStr("    Node_ID = ", rd->Node_ID,
sizeof(Qcst_Node_Id_t));
    printf("%s %d \n", "      Node_Role = ", rd->Node_Role);
    printf("%s", "      Membership_Status = ");
    switch (rd->Membership_Status) {
      case 0: str = "Active";
              break;
      case 1: str = "Inactive";
              break;
      case 2: str = "Partition";
              break;
      default: str = "unknown node status";
    }
    printf("%s \n", str);
    rd++;
  }
  return;
}  /* end printRcvyDomain                                                   */

/**************************************************************************/
/*                                                                        */
/* Concatenate a null terminated string and a non null terminated string  */
/* and print it.                                                          */
/*                                                                        */
/**************************************************************************/
static void printStr(char *s1, char *s2, unsigned int len) {

  char buffer[132];
```

```
|   memset(buffer, 0x00, sizeof(buffer));
|   memcpy(buffer, s1, strlen(s1));
|   strncat(buffer, s2, len);
|   printf("%s \n", buffer);
|   return;
| }  /* end printStr                                              */
|
```

Planning for clusters

Find out what you need to do before you can set up clusters on your System i products. Find out the prerequisites for clusters as well as hints on designing your cluster. Finally, read tips for setting up your network and some performance hints for clusters.

This topic covers the requirements that you will need before you can use clustering. The following topics provide you with the general concepts, requirements, and considerations for designing a clustering solution.

Solutions for configuring and managing clusters

Cluster resource services provides the basic cluster infrastructure. There are several methods that will allow you to take advantage of the clustering capabilities provided by cluster resource services.

i5/OS cluster resource services on the System i product provides the basic infrastructure that allows you to use a cluster. Cluster resource services provides a set of integrated services that maintain cluster topology, perform heartbeating, and allow creation and administration of cluster configuration and cluster resource groups. Cluster resource services also provides reliable messaging functions that keep track of each node in the cluster and ensure that all nodes have consistent information about the state of cluster resources.

While cluster resource services provides the basic cluster infrastructure, there are several methods that will allow you to take advantage of these clustering capabilities. Each has distinct benefits and capabilities.

Important: Use only one of these solutions exclusively. Conflicts, problems and unpredictability can arise when attempting to use more than one solution to create and manage a cluster. The information you find in the i5/OS Information Center documents procedures specific to iSeries Navigator and the cluster resource services CL commands and APIs. If you use a cluster middleware IBM Business Partner solution, see the documentation provided with the product for procedural information on performing tasks.

iSeries Navigator cluster management

IBM offers a cluster management interface that is available through iSeries Navigator and accessible through Option 41 (i5/OS - HA Switchable Resources).

| This interface allows you to create and manage a cluster that uses switchable independent disk pools
| (switchable independent ASPs) to ensure data availability. It also allows you to create and manage
| clusters, CRGs, cluster administrative domains, and resources.

Important: The iSeries Navigator cluster management interface does not contain all of the capabilities provided by cluster resource services. While iSeries Navigator provides many functions necessary to configure and manage a cluster, be aware that there are some capabilities that are only available through the cluster commands and APIs, or perhaps through a cluster middleware IBM Business Partner application, depending on the particular application. For example, the System i clustering architecture supports up to 128 nodes in a cluster, however the iSeries Navigator interface only supports up to four nodes in a cluster. iSeries Navigator cluster management features a wizard which steps you through the creation of a simple, four-node cluster. If your clustering needs exceed this, you should consider using the IBM cluster commands and APIs or cluster middleware IBM Business Partner products.

| You can also use iSeries Navigator to perform other cluster-related tasks. Some of these tasks include:

- Adding a node to an existing cluster
- Adding a switchable devices to a cluster
- Adding a switchable applications to a cluster
- Adding a switchable data group to a cluster
- Change the role of nodes in the recovery domain
- Changing the cluster description
- Deleting a cluster
- Starting clustering
- Stopping clustering
- Viewing messages about cluster activity
- | Creating a cluster administrative domain
- | Adding a monitored resource entry

| For a complete list cluster-related tasks that are available in iSeries Navigator, see the online help for
| clusters.

Note: The iSeries Navigator cluster management interface does not support logical object replication. For replication, you should consider the clustering products available from high availability IBM Business Partners.

Related concepts

iSeries Navigator

"Cluster commands and APIs"
i5/OS cluster resource services provides a set of control language (CL) commands, application program interfaces (APIs) and facilities that can be used by System i application providers or customers to enhance their application availability.

"Cluster middleware IBM Business Partners and available clustering products" on page 91
You can purchase a product from an IBM cluster middleware IBM Business Partner that provides the logical replication functions that are integral to clustering and simplifies cluster creation and management.

"Common cluster problems" on page 152
Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

Related reference

"Frequently asked questions about iSeries Navigator cluster management" on page 161
Questions and answers about the iSeries Navigator graphical user interface for creating and managing clusters.

Cluster commands and APIs

i5/OS cluster resource services provides a set of control language (CL) commands, application program interfaces (APIs) and facilities that can be used by System i application providers or customers to enhance their application availability.

You can write your own custom application to configure and manage your cluster by utilizing cluster control language (CL) commands and application programming interfaces (APIs). These commands and APIs take advantage of the technology provided by cluster resource services provided as a part of i5/OS.

QUSRTOOL
Cluster resource services also provides a set of example commands in the QUSRTOOL library

that map to the APIs that have no supported command interface. The QUSRTOOL commands might be useful in some environments. For example, you can change heartbeating or send information around the cluster. See the member TCSTINFO in the file QUSRTOOL/QATTINFO for more information about these example commands. An example application CRG exit program is also included in the QUSRTOOL library. The sample source code can be used as the basis for writing an exit program. Sample source, TCSTDTAEXT, in file QATTSYSC contains a source for a program to create the QCSTHAAPPI and QCSTHAAPP0 dataareas, and QACSTOSDS (object specifier) file.

Related tasks

"Adding a node to a cluster" on page 113
You can add a node to a cluster using iSeries Navigator or commands.

Cluster CL command and API descriptions:

There are many APIs and CL commands that you can use for configuring, activating, and managing clusters, cluster nodes, and cluster resource groups.

The following tables show the name and a brief description of the cluster control and the cluster resource group control language (CL) commands and APIs that are available. The cluster CL commands are only available on OS/400 V5R2M0 or later.

| The first table contains commands and APIs for configuring, activating, and managing a **cluster and the nodes** in a cluster. The second table contains commands and APIs for configuring, activating, and managing **cluster resource groups** in a cluster. The third table contains commands for configuring and managing a **cluster administrative domain**. The fourth table contains descriptions of Integrated Operating System API which can be used to add and remove monitored resource entries from a cluster administrative domain.

Table 17. Cluster Control CL Command and API Descriptions

Description	Cluster Control CL Command	Cluster Control API Name
Add Cluster Node Entry Adds a node to the membership list of an existing cluster. Also assigns the IP interface addresses to be used by cluster communications.	ADDCLUNODE	Add Cluster Node Entry (QcstAddClusterNodeEntry)
Add Device Domain Entry Adds a node to a device domain membership list so that it can participate in recovery actions for resilient devices. The addition of the first node to a device domain has the effect of creating that device domain.	ADDDEVDMNE	Add Device Domain Entry (QcstAddDeviceDomainEntry)
Adjust Cluster Version, Change Cluster Version Adjusts the current cluster version to the next level, for example, so that new function can be used within the cluster.	CHGCLUVER	Adjust Cluster Version (QcstAdjustClusterVersion)

Table 17. Cluster Control CL Command and API Descriptions (continued)

Description	Cluster Control CL Command	Cluster Control API Name
Change Cluster Node Entry Changes the fields in the cluster node entry. For example, the IP interface addresses used for cluster communications can be changed.	CHGCLUNODE	Change Cluster Node Entry (QcstChangeClusterNodeEntry)
Change Cluster Resource Services, Change Cluster Configuration Adjusts cluster performance and configuration tuning parameters to match the communications environment of the network used for cluster communications.	CHGCLUCFG	Change Cluster Resource Services (QcstChgClusterResourceServices)
Create Cluster Creates a new cluster of one or more nodes.	CRTCLU	Create Cluster (QcstCreateCluster)
Delete Cluster Deletes an existing cluster. Cluster resource services is ended on all active cluster nodes and they are removed from the cluster.	DLTCLU	Delete Cluster (QcstDeleteCluster)
End Cluster Node Ends cluster resource services on one or all nodes in the membership list of an existing cluster. The node becomes unavailable to the cluster until it is restarted using the Start Cluster Node interface.	ENDCLUNOD	End Cluster Node (QcstEndClusterNode)
List Cluster Information, Display Cluster Information Retrieves information about a cluster. For example, the complete cluster membership list can be returned.	DSPCLUINF	List Cluster Information (QcstListClusterInfo)
List Device Domain Information, Display Cluster Information Lists device domain information of a cluster. For example, the list of currently defined device domains can be returned.	DSPCLUINF	List Device Domain Information (QcstListDeviceDomainInfo)
Remove Cluster Node Entry Removes a node from the membership list of a cluster. The node is removed from any recovery domains, cluster operations are ended on the node, and all cluster resource services objects are deleted from the node.	RMVCLUNODE	Remove Cluster Node Entry (QcstRemoveClusterNodeEntry)

Table 17. Cluster Control CL Command and API Descriptions (continued)

Description	Cluster Control CL Command	Cluster Control API Name
Remove Device Domain Entry Removes a node from a device domain membership list. If this is the last node in the device domain, this also has the effect of deleting the device domain from the cluster.	RMVDEVDMNE	Remove Device Domain Entry (QcstRemoveDeviceDomainEntry)
Retrieve Cluster Information, Display Cluster Information Retrieves information about a cluster. For example, the cluster name and current cluster version are returned.	DSPCLUINF	Retrieve Cluster Information (QcstRetrieveClusterInfo)
Retrieve Cluster Resource Services Information, Display Cluster Information Retrieves information about the cluster resource services performance tuning and configuration parameters.	DSPCLUINF	Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo)
Start Cluster Node Starts cluster resource services on a node that is part of a cluster but is currently not active.	STRCLUNOD	Start Cluster Node (QcstStartClusterNode)
Work with Cluster Displays and works with cluster nodes and objects.	WRKCLU	none

Table 18. Cluster Resource Group CL Command and API Descriptions

Description	Cluster Resource Group CL Command	Cluster Resource Group API
Add Cluster Resource Group Device Entry Adds a new device entry to a cluster resource group. The device becomes a member of the group of switchable devices.	ADDCRGDEVE	Add Cluster Resource Group Device Entry (QcstAddClusterResourceGroupDevice)
Add Node to Recovery Domain, Add Cluster Resource Group Node Entry Adds a new node to the recovery domain of an existing cluster resource group.	ADDCRGNODE	Add Node to Recovery Domain (QcstAddNodeToRcvyDomain)
Change Cluster Resource Group Changes attributes of a cluster resource group. For example, the takeover IP address for an application CRG can be modified.	CHGCRG	Change Cluster Resource Group (QcstChangeClusterResourceGroup)

Table 18. Cluster Resource Group CL Command and API Descriptions (continued)

Description	Cluster Resource Group CL Command	Cluster Resource Group API
Change Cluster Resource Group Device Entry Changes a device entry in a cluster resource group. For example, the option to vary the configuration object online at switchover or failover can be modified.	CHGCRGDEVE	Change Cluster Resource Group Device Entry (QcstChangeClusterResourceGroupDev)
Create Cluster Resource Group Creates a cluster resource group object. The cluster resource group object identifies a recovery domain, which is a set of nodes in the cluster that will play a role in recovery.	CRTCRG	Create Cluster Resource Group (QcstCreateClusterResourceGroup)
Delete Cluster Resource Group Deletes a cluster resource group (CRG) on the local node only. Deleting a local cluster resource group requires the Cluster Resource Services to be inactive.	DLTCRG	none
Delete Cluster Resource Group, Delete CRG Cluster Deletes a cluster resource group from the cluster. The CRG object will be deleted from all active nodes in the recovery domain.	DLTCRGCLU	Delete Cluster Resource Group (QcstDeleteClusterResourceGroup)
Distribute Information Delivers information from a node in the recovery domain of a CRG to other nodes in that CRG's recovery domain.	none	Distribute Information (QcstDistributeInformation)
End Cluster Resource Group Disables resiliency of the specified cluster resource group. Upon successful completion of this API, the cluster resource group status is set to inactive.	ENDCRG	End Cluster Resource Group (QcstEndClusterResourceGroup)
Initiate Switchover, Change Cluster Resource Group Primary Causes an administrative switchover to be performed for the cluster resource group. The recovery domain is changed so that the current primary node becomes the last backup and the current first backup node becomes the new primary.	CHGCRGPRI	Initiate Switchover (QcstInitiateSwitchover)

Table 18. Cluster Resource Group CL Command and API Descriptions (continued)

Description	Cluster Resource Group CL Command	Cluster Resource Group API
List Cluster Resource Groups, Display Cluster Resource Group Information Generates a list of cluster resource groups and some information about the cluster resource group in the cluster.	DSPCRGINF	List Cluster Resource Groups (QcstListClusterResourceGroups)
List Cluster Resource Group Information, Display Cluster Resource Group Information Returns the contents of a cluster resource group object. For example, the recovery domain with the current node roles can be returned.	DSPCRGINF	List Cluster Resource Group Information (QcstListClusterResourceGroupInf)
Remove Cluster Resource Group Device Entry Removes a device entry from a cluster resource group. The device will no longer be a switchable resource.	RMVCRGDEVE	Remove Cluster Resource Group Device Entry (QcstRemoveClusterResourceGroupDev)
Remove Node From Recovery Domain, Remove Cluster Resource Group Node Entry Removes a node from the recovery domain of an existing cluster resource group. The node will no longer participate in recovery action for that group of resources.	RMVCRGNODE	Remove Node From Recovery Domain (QcstRemoveNodeFromRcvyDomain)
Start Cluster Resource Group Enables resiliency for the specified cluster resource group. The cluster resource group becomes active within the cluster.	STRCRG	Start Cluster Resource Group (QcstStartClusterResourceGroup)

Note: Cluster resource services also provides a set of example commands in the QUSRTOOL library that map to the CL commands and APIs mentioned above. The QUSRTOOL commands might be useful in some environments. For example, one can easily set up a cluster for testing cluster-enabled applications. See the member TCSTINFO in the file QUSRTOOL/QATTINFO for more information about these example commands.

Table 19. Administrative Domain CL Command Descriptions

Description	Administrative Domain CL Command	Administrative Domain APIs
Create Administrative Domain Creates a peer CRG which represents a cluster administrative domain. Once the cluster administrative domain is created, monitored resource entries (MREs), can be added to the domain to synchronize resource changes. **Note:** Once the cluster administrative domain is created you can use the CRG commands in Table 18 on page 87 to manage it.	CRTADMDMN	None
Delete Administrative Domain Deletes the peer CRG representing the cluster administration domain. Upon completion all MREs are removed from the domain and changes to the resources that were being monitored will no longer be propagated.	DLTADMDMN	None

Table 20. Integrated Operating System API descriptions. In addition to these cluster administrative domain CL commands, there are also several Integrated Operating System application programmable interfaces (APIs) that provide the ability to add and remove monitored resource entries.

Description	CL Commands[1]	Integrated Operating System API
Add Monitored Resource Entry Adds a monitored resource entry for a system resource and its attributes.	None	Add Monitored Resource Entry (QfpadAddMonitoredResourceEntry)
Remove Monitored Resource Entry Removes an Monitored Resource Entry (MRE) from the monitored resource directory.	None	Remove Monitored Resource Entry (QfpadRmvMonitoredResourceEntry)
Retrieve Monitored Resource Information Returns information about monitored resources.	None	Retrieve Monitored Resource Information (QfpadRtvMonitoredResourceInfo)
Note: 1. There is no supported CL command equivalent for this function. The source for a non-supported command and call processing program (CPP) has been provided in the QUSRTOOL library. To learn about this command source and CPP, look at the member TFPADINFO in file QATTINFO.		

Related reference

Cluster APIs

Cluster middleware IBM Business Partners and available clustering products

You can purchase a product from an IBM cluster middleware IBM Business Partner that provides the logical replication functions that are integral to clustering and simplifies cluster creation and management.

IBM cluster middleware IBM Business Partners provide software solutions for dedicated replication and cluster management functions. If you want to purchase a product that provides logical replication functions that are integral to clustering and simplifies cluster creation and management, contact your IBM marketing representative or business partner. They can provide a complete list of clustering enabling products provided by IBM cluster middleware IBM Business Partners.

The cluster middleware IBM Business Partner cluster management product:

- Provides the user interface to define and maintain the cluster configuration
- Provides the user interface to define and manage the device, data, and application cluster resource groups
- Maintains the knowledge through the use of cluster APIs, of what cluster resource groups are defined in the cluster and what relationships are required.
- Creates the device, data, and application cluster resource groups.

The cluster middleware IBM Business Partner replication product:

- Builds the middleware's control structures that identify the data and objects that are required to be resilient.
- Creates the cluster resource group for critical data and associates that object with its control structures.
- Provides the exit program for the data cluster resource group.

Related tasks

"Adding a node to a cluster" on page 113
You can add a node to a cluster using iSeries Navigator or commands.

Cluster requirements

Outlines the hardware, software, and communications requirements for using clusters.

The requirements for using clusters vary depending upon which cluster capabilities you choose to use. For example, you may choose to use a simple, two-node cluster to take advantage of logical replication. Or you may choose to use a cluster designed to take advantage of switched disks and switchable independent disk pools.

Related concepts

"Examples: Cluster configurations" on page 134
Use these examples of typical cluster implementations to understand when, why, and how using clusters can be beneficial.

Hardware requirements for clusters

Any System i model that is capable of running OS/400 V4R4, or later, is compatible for using clustering.

In addition, you should provide protection from a power loss through an external uninterruptible power supply or equivalent. Otherwise, a sudden power loss on a cluster node can result in a cluster partition state rather than a failover.

Clustering uses Internet Protocol (IP) multicast capabilities. Multicast does not map well to all types of physical media. For more information about multicast restrictions that may apply to your particular hardware, see the TCP/IP Setup.

If you plan to use independent disk pools in your cluster, refer to the Hardware requirements for independent disk pools topic. You can also protect your disks with mirrored protection or device parity protection. Using these solutions on your primary system prevents a failover from occurring should a protected disk fail. It also is a good idea to have these solutions on your backup system in case a failover should occur. See Disk protection for details.

| **Note:** For details on other requirements that are necessary before configuring clusters, see "Cluster
| configuration checklist" on page 105.

Related concepts

Uninterruptible power supply

"Cluster partition" on page 37
A *cluster partition* is a subset of the active cluster nodes that results from a communications failure. Members of a partition maintain connectivity with each other.

"Failover" on page 26
Failover occurs when a server in a cluster automatically switches over to one or more backup servers in the event of a system failure.

Software and license requirements for clusters

In order to use clustering, you must have the correct software and licenses.

1. OS/400 V4R4M0 or later configured with TCP/IP (TCP/IP Connectivity Utilities).
2. A cluster configuration and management software solution. This can be any of the following:
   - iSeries Navigator cluster management
   - A cluster middleware IBM Business Partner solution
   - Your own cluster management application program written using cluster resource services commands and APIs
| 3. See the "Cluster configuration checklist" on page 105

Important: If you plan to use independent disk pools to take advantage of switchable devices, there are additional requirements. See Plan for independent disk pools for details.

Related concepts

"Solutions for configuring and managing clusters" on page 83
Cluster resource services provides the basic cluster infrastructure. There are several methods that will allow you to take advantage of the clustering capabilities provided by cluster resource services.

"Cluster version" on page 20
A *cluster version* represents the level of function available on the cluster.

Communications requirements for clusters

Use any type of communications media in your clustering environment as long as it supports Internet Protocol (IP).

| Cluster resource services uses TCP/IP and UDP/IP protocols to communicate between nodes. Local area
| networks (LANs), wide area networks (WANs), OptiConnect system area networks (SANs), or any
| combination of these connectivity devices are supported. Your choice should be based on the following
| factors:

- Volume of transactions
- Response time requirements
- Distance between the nodes
- Cost considerations

You can use these same considerations when determining the connection media to be used to connect primary and backup locations of resources. When planning your cluster, it is recommended that you designate one or more of your backup nodes in remote locations in order to survive a site loss disaster.

To avoid performance problems that might be caused by inadequate capacity, you need to evaluate the communication media that is used to handle the volumes of information that is sent from node to node. You can choose which physical media you prefer to use such as token ring, Ethernet, asynchronous transfer mode (ATM), SPD OptiConnect, High-Speed Link (HSL) OptiConnect, or Virtual OptiConnect (a high-speed internal connection between logical partitions).

HSL OptiConnect is a technology provided by OptiConnect for i5/OS software (i5/OS Option 23 - i5/OS OptiConnect). It can be used to construct highly available solutions. HSL OptiConnect is a system area network that provides high-speed, point-to-point connectivity between cluster nodes by using High Speed Link (HSL) Loop technology. HSL OptiConnect requires standard HSL cables, but no additional hardware.

For switchable hardware, also referred to as resilient device CRGs, you need to have an independent disk pool that is switchable in your environment. In a logical partition environment, this is a collection of disk units that are on the bus that is being shared by the logical partitions, or that are attached to an input/output processor that has been assigned to an I/O pool. For a multi-system environment, this is one or more switchable expansion units (towers) properly configured on the HSL loop also containing the systems in the recovery domain. The switchable tower can also be used in an LPAR environment. For more planning information about switchable hardware and independent disk pools, see Plan for independent disk pools.

Note: If you are using 2810 LAN adapters using **only** TCP/IP, and not using Systems Network Architecture (SNA) or IPX, you can increase your adapter performance on an OS/400 V4R5M0 system by specifying Enable only for TCP(*YES) for your specific line description using the Work with Line Descriptions (WRKLIND) command. Enable only for TCP(*YES) is set automatically in OS/400 V5R1M0 and later releases.

Related information

OptiConnect for i5/OS

Designing your cluster

Identify your needs to determine how to design your cluster.

Because there are a variety of ways to use clustering depending on what you are hoping to achieve, it is important to spend some time identifying your needs to determine how to design your cluster.

Designing your network for clusters

Before you configure your networks for clustering, you need to plan carefully and do some initial pre-cluster configuration involving TCP/IP.

It is important that you read these topics before configuring your cluster. They will tell you when or how to:

- Set up IP addresses
- Set TCP/IP configuration attributes
- Avoid a cluster partition

For information about setting up redundant communications paths and whether you need to have a dedicated network for clustering, see Dedicate a network for clusters.

Setting up IP addresses:

Because cluster resource services uses *only* IP to communicate with other cluster nodes, all cluster nodes must be *IP-reachable*.

This means that you must have IP interfaces configured to connect the nodes in your cluster. These IP addresses must be set up either manually by the network administrator in the TCP/IP routing tables on each cluster node or they may be generated by routing protocols running on the routers in the network. This TCP/IP routing table is the map that clustering uses to find each node; therefore, each node must have its own *unique* IP address. Each node may have up to two IP addresses assigned to it. These addresses must not be changed in any way by other network communications applications. Be sure when you assign each address that you take into account which address uses which kind of communication line. If you have a preference for using a specific type of communication media, make sure to configure the first IP address using your preferred media. The first IP address is what is treated preferentially by the reliable message function and heartbeat monitoring. All IP addresses on a node must be able to reach every other IP address in the cluster. One address can reach another address if you can ping and use a UDP message traceroute in both directions.

Note: You need to be sure that the loop back address (127.0.0.1) is active for clustering. This address, which is used to send any messages back to the local node, is normally active by default. However, if it has been ended by mistake, cluster messaging cannot function until this address has been restarted.

Related concepts

"Reliable message function" on page 36
The *reliable message function* of cluster resource services keeps track of each node in a cluster and ensures that all nodes have consistent information about the state of cluster resources.

"Heartbeat monitoring" on page 35
Heartbeat monitoring is a cluster resource services function that ensures that each node is active by sending a signal from every node in the cluster to every other node in the cluster to convey that they are still active.

Setting TCP/IP configuration attributes:

To enable cluster resource services, certain attribute settings are required in the TCP/IP configuration of your network.

You must set these attributes before you can add any node to a cluster:
- Set IP datagram forwarding to *YES using the CHGTCPA (Change TCP/IP Attributes) command if you plan to use a System i product as the router to communicate with other networks and you have no other routing protocols running on that server.
- Set the INETD server to START. See INETD server for information about starting the INETD server.
- Set User Datagram Protocol (UDP) CHECKSUM to *YES using the CHGTCPA (Change TCP/IP Attributes) command.
- Set MCAST forwarding to *YES if you are using bridges to connect your token ring networks.
- If you are using OptiConnect for i5/OS to communicate between cluster nodes, start the QSOC subsystem by specifying STRSBS(QSOC/QSOC).

Starting the INETD server:

The Internet daemon (INETD) server must be started in order for a node to be added or started, as well as for merge partition processing.

It is recommended that the INETD server always be running in your cluster.

Using iSeries Navigator

This requires Option 41 (i5/OS - HA Switchable Resources) to be installed and licensed.

To start the INETD server, follow these steps:

1. In iSeries Navigator, expand **Network**.
2. Expand *Servers*.
3. Expand **TCP/IP**.
4. Right-click **INETD** and select **Start**.

Using CL commands and APIs

The INETD server can also be started using the STRTCPSVR (Start TCP/IP Server) command and specifying the *INETD parameter. When the INETD server is started, a QTOGINTD (User QTCP) job will be present in the Active Jobs list on the subject node.

Related concepts

"Common cluster problems" on page 152
Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

Related reference

STRTCPSVR (Start TCP/IP Server) command

Tips: Cluster communications:

Consider these tips when you set up your communications paths.

- Be sure you have adequate bandwidth on your communication lines to handle the non-cluster activity along with the clustering heartbeating function and continue to monitor for increased activity.
- For best reliability, do not configure a single communication path linking one or more nodes.
- Do not overburden the line that is responsible for ensuring that you are still communicating with a node.
- Eliminate as many single points of failure as possible such as having two communication lines coming into a single adapter, same input-output processor (IOP), or same tower.
- If you have an extremely high volume of data being passed over your communication lines, you may want to consider putting data replication and heartbeat monitoring on separate networks.
- If you are using Internet Protocol (IP) multicast, you should see the TCP/IP Configuration and Reference for multicast restrictions that may apply to different types of physical media.
- User Datagram Protocol (UDP) multicast is the preferred protocol that the cluster communications infrastructure uses to send cluster management information between nodes in a cluster. When the physical media supports multicast capabilities, cluster communications utilizes the UDP multicast to send management messaging from a given node to all local cluster nodes that support the same subnet address. Messages that are sent to nodes on remote networks are always sent using UDP point-to-point capabilities. Cluster communications does not rely on routing capability for multicast messages.
- The multicast traffic that supports cluster management messaging tends to fluctuate by nature. Depending on the number of nodes on a given LAN (that supports a common subnet address) and the complexity of the cluster management structure that is chosen by the cluster administrator, cluster related multicast packets can easily exceed 40 packets per second. Fluctuations of this nature can have a negative impact on older networking equipment. One example is congestion problems on devices on the LAN that serve as Simple Network Management Protocol (SNMP) agents that need to evaluate every UDP multicast packet. Some of the earlier networking equipment does not have adequate bandwidth to keep up with this type of traffic. You need to ensure that you or the network administrator has reviewed the capacity of the networks to handle UDP multicast traffic to make certain that clustering does not have a negative impact on the performance of the networks.

Related concepts

"Planning for logical replication" on page 102
Multiple copies of the data are maintained with logical replication. Data is replicated, or copied, from

the primary node in the cluster to the backup nodes designated in the recovery domain. When an outage occurs on the primary node, the data remains available as a designated backup node takes over as the primary point of access.

"Reliable message function" on page 36
The *reliable message function* of cluster resource services keeps track of each node in a cluster and ensures that all nodes have consistent information about the state of cluster resources.

Related information

TCP/IP setup

Avoiding a cluster partition:

The typical network-related cluster partition can best be avoided by configuring redundant communications paths between all nodes in the cluster.

A *redundant communications path* means that you have two lines configured between two nodes in a cluster. If a failure on the first communication path should occur, the second communication path can take over to keep communications running between the nodes, thereby minimizing conditions that can put one or more nodes of the cluster into a cluster partition. One thing to consider when configuring these paths is that if both of your communications lines go into the same adapter on the system, these lines are still at risk if this single adapter fails.

| However, it should be noted that a cluster partition is not always avoidable. If your system experiences a
| power loss or if a hardware failure occurs, the cluster may become partitioned.

Related concepts

"Cluster partition" on page 37
A *cluster partition* is a subset of the active cluster nodes that results from a communications failure. Members of a partition maintain connectivity with each other.

"Tips: Cluster communications" on page 95
Consider these tips when you set up your communications paths.

"Partition errors" on page 154
Certain cluster conditions are easily corrected. If a cluster partition has occurred, you can learn how to recover. This topic also tells you how to avoid a cluster partition and gives you an example of how to merge partitions back together.

Dedicating a network for clusters:

During normal operations, base clustering communication traffic will be minimal. It is, however, highly recommended that you have redundant communication paths configured for each node in a cluster.

By configuring two lines, you can dedicate one line for clustering traffic and the other line can handle the normal traffic and also be the backup line if the dedicated line for clustering goes down.

Related concepts

"Avoiding a cluster partition"
The typical network-related cluster partition can best be avoided by configuring redundant communications paths between all nodes in the cluster.

Multiple-release clusters

If creating a cluster that will include nodes at multiple cluster versions, then certain steps are required when you create the cluster.

By default, the current cluster version will be set to the potential cluster version of the first node added to the cluster. This approach is appropriate if this node is at the lowest version level to be in the cluster. However, if this node is at a later version level, then you will subsequently be unable to add nodes with a lower version level. The alternative is to use the target cluster version value on create cluster to set the current cluster version to one less than the potential cluster version of the first node added to the cluster.

For example, consider the case where a two-node cluster is to be created. The nodes for this cluster are:

Node Identifier	Release	Potential Cluster Version
Node A	V5R3	4
Node B	V5R4	5

If the cluster is to be created from Node B, care must be taken to indicate that this will be a mixed release cluster. The target cluster version must be set to indicate that the nodes of the cluster will communicate at one less than the requesting node's potential node version.

Related concepts

"Cluster version" on page 20
A *cluster version* represents the level of function available on the cluster.

Identifying systems to include in a cluster

To identify the systems that you want to include in a cluster, you need to decide which systems are capable of providing adequate backup for the data and applications that you need to run your business.

You need to determine:
- Which systems contain your critical data and critical applications?
- Which systems will be the backup for those systems?

Once you have determined this, these are the systems that you will want to include in your cluster.

Comparison between primary-backup and peer models

Primary-backup CRGs and peer CRGs provide resiliency for resources within a cluster; however, it is important to understand their differences and uses.

Clusters support two models for defining CRGs in your environment. Roles are defined for both primary-backup and peer models. In primary-backup model, they need to define an order. Nodes that are defined as backup nodes provide access to resources on the primary node in the event of a node failure. With the peer model, each node are equal in the role and can provide access for the resource; however there is no concept of order.

Primary-backup model

With the primary-backup model, users must define the node as either a either primary, backup, or replicate role. These roles are defined and managed within the recovery domain. If a node has been defined as the primary access point for the resource, then other nodes provide backup if the primary node fails.

Peer model

Peer model CRGs eliminate the need to define an ordered recovery domain. For a peer model, nodes can be defined as either peer or replicate. If nodes defined as peer, then all the nodes in the recovery domain are equal and can provide the access point for the resource.

Identifying applications to include in a cluster

Not every application will give you the availability benefits of clustering.

An application must be resilient in order to take advantage of the switchover and failover capabilities provided by clustering. Application resilience allows the application to be restarted on the backup node without having to reconfigure the clients using the application. Therefore your application must meet certain requirements to take full advantage of the capabilities offered by clustering. See the Cluster applications topic for more information about resilient applications.

Planning for data resilience

Data resilience is achieved when data is always available to an user or application. You can achieve data resilience through the use of logical replication or switchable independent disk pools.

Determining which data should be made resilient:

Understand what types of data you should consider making resilient.

Determining which data you need to make resilient is similar to determining which kind of data you need to backup and save when you prepare a back up and recovery strategy for your systems. You need to determine which data in your environment is critical to keeping your business up and running.

For example, if you are running a business on the Web, your critical data may be:
- Today's orders
- Inventory
- Customer records

In general, information that does not change often or that you do not need to use on a daily basis probably does not need to be made resilient.

Related concepts
Planning a backup and recovery strategy

Comparison of logical replication, switched disks, and cross-site mirroring:

This topic provides an overview of different data resilience technologies that can be used with clusters to enhance high availability.

Data resiliency allows data to remain available to applications and users even though the system that originally hosted the data fails. Choosing the correct set of data resiliency technologies in the context of your overall business continuity strategy can be complex and difficult. It's important to understand the different data resilience solutions that can be used to enhance availability in multiple system environments. You can either choose a single solution or use a combination of these technologies to meet your needs.

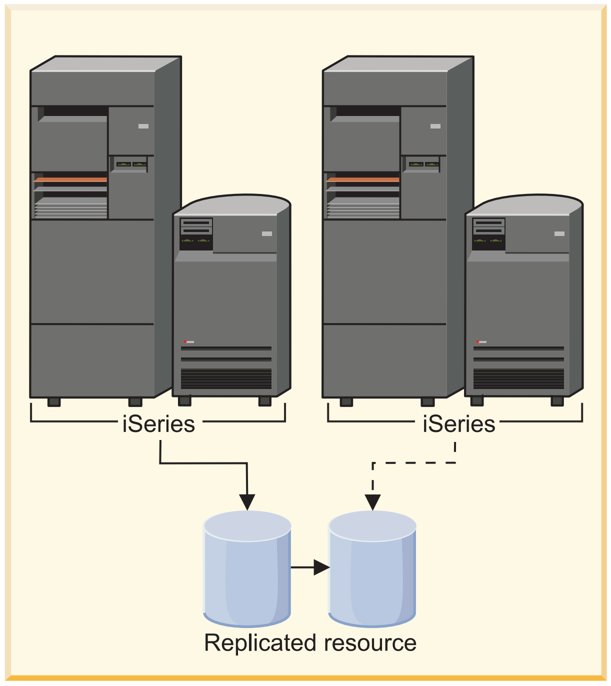
For more details on these solutions, see Data Resilience Solutions for IBM i5/OS High Availability Clusters. The section called "Data Resilience Solutions for IBM i5/OS High Availability Clusters" contains a detailed comparison of the attributes for each of these technologies.

Logical replication

Logical replication is the process of copying objects from one node in a cluster to one or more other nodes in the cluster, which makes the objects on all the systems identical.

A replicated resource allows for objects, such as an application and its data, to be copied from one node in the cluster to one or more other nodes in the cluster. This process keeps the objects on all servers in the resource's recovery domain identical. If you make a change to an object on one node in a cluster, the change is replicated to other nodes in the cluster. Then, should a failover or switchover occur, the backup node can seamlessly take on the role of the primary node. The server or servers that act as backups are defined in the recovery domain. When an outage occurs on the server defined as the primary node in the recovery domain and a switchover or failover is initiated, the node designated as the backup in the recovery domain becomes the primary access point for the resource.

The following graphic shows data from one node being replicated to another node.



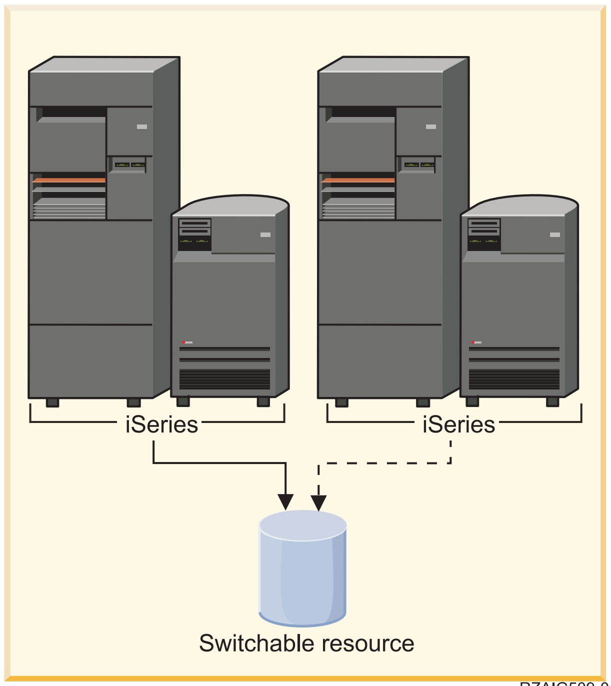
RZAIG508-0

Replication requires the use of either a custom-written application or a software application written by a cluster middleware business partner. See Plan for logical replication for details.

Switchable disks

Switchable disks enable the resources, such as data and applications, residing on an expansion unit or on an input-output processor (IOP) on a shared bus or in an I/O Pool for a logical partition, to be switched between a cluster's primary node and backup node. This allows for a set of disk units to be accessed from a second server, a server defined as a backup node in the cluster resource group's recovery domain, when the server currently using those disk units experiences an outage and a failover or switchover occurs.

The following graphic shows how a resource is switched between a nodes.



Taking advantage of switchable resources in your cluster requires the use of independent disk pools. See Plan for independent disk pools for more information.

Cross-site mirroring

Cross-site mirroring, combined with the geographic mirroring function, enables you to mirror data on disks at sites that can be separated by a significant geographic distance. This technology can be used to extend the functionality of a device cluster resource group (CRG) beyond the limits of physical component connection. Geographic mirroring provides the ability to replicate changes made to the production copy of an independent disk pool to a mirror copy of that independent disk pool. As data is written to the production copy of an independent disk pool, the operating system mirrors that data to a second copy of the independent disk pool through another system. This process keeps multiple identical copies of the data.

Through the device CRG, should a failover or switchover occur, the backup node can seamlessly take on the role of the primary node. The server or servers that act as backups are defined in the recovery domain. The backup nodes can be at the same or different physical location as the primary. When an outage occurs on the server defined as the primary node in the recovery domain and a switchover or failover is initiated, the node designated as the backup in the recovery domain becomes the primary access point for the resource and will then own the production copy of the independent disk pool. Thus, you can gain protection from the single point of failure associated with switchable resources.

| The following graphic shows an example of geographic mirroring used in a clustered environment.

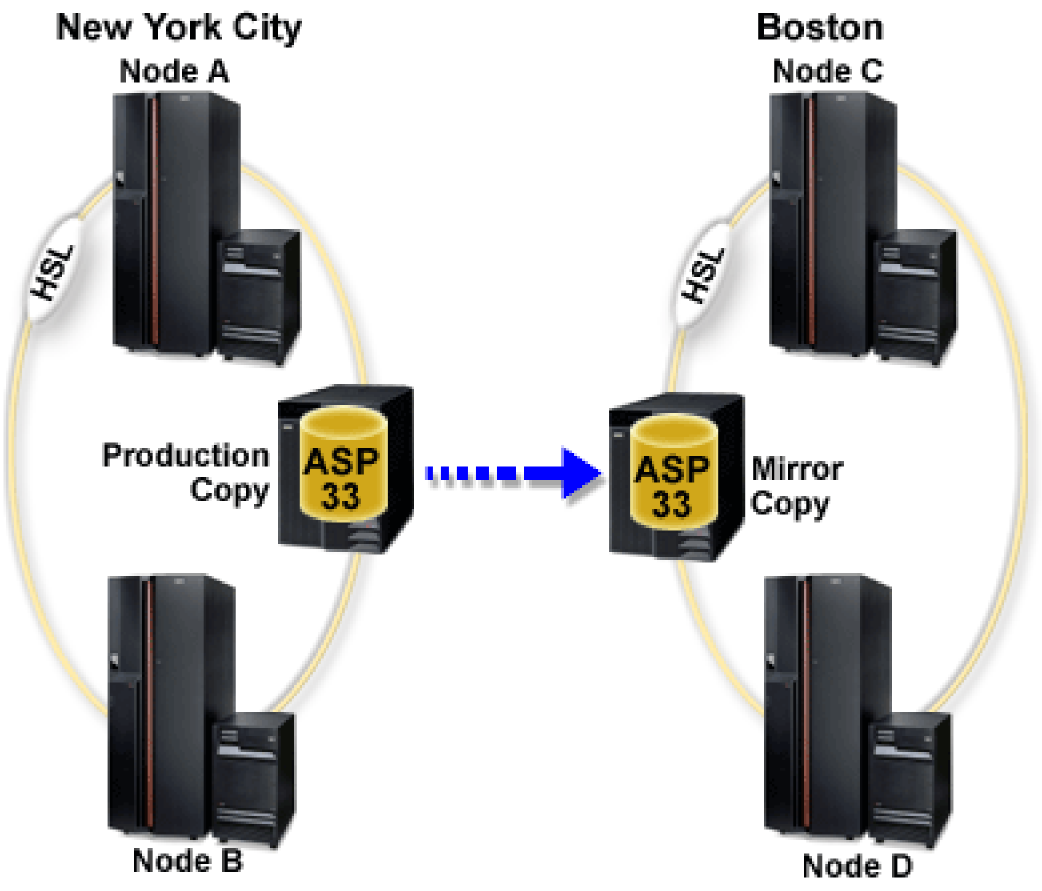


Table 21. Comparison of data resilient technologies that can be used with clusters. Learn about characteristics of different data resiliency technologies to help you determine the best solution for your cluster.

Factor	**Replication**	**Switchable disks**	**Cross-site mirroring**
Flexibility	10s of servers	2 servers	4 servers
Single point of failure	None	Disk subsystem	None
Cost	• Additional disk capacity required. • Business partner replication software. • Duplicate disks	• Switchable I/O expansion tower or IOP • Option 41	• Additional disk for mirror copy of independent disk pool • Optionally switchable I/O expansion unit • Option 41
Performance	Replication overhead	Little impact	Geographic mirroring overhead
Real time coverage	Journaled objects	Objects contained in independent disk pool	Objects contained in independent disk pool
Geographic dispersion	Limited by performance considerations	Limited attach distance as servers and expansion units must be attached to HSL OptiConnect loop (250 meters maximum)	Limited by performance considerations (No limits are imposed by the system. However, response time and throughput over selected communications lines can dictate some practical limit.)

Table 21. Comparison of data resilient technologies that can be used with clusters (continued). Learn about characteristics of different data resiliency technologies to help you determine the best solution for your cluster.

Factor	Replication	Switchable disks	Cross-site mirroring
Disaster recovery protection	Yes	No	Yes
Concurrent backup	Yes	No	No
Setup	• Replication environment. • Determining what to replicate.	• Independent disk pool environment. • Populate independent disk pool	• Independent disk pool environment (include setting up geographic mirroring) • Populate independent disk pool

Related concepts

"Planning for logical replication"
Multiple copies of the data are maintained with logical replication. Data is replicated, or copied, from the primary node in the cluster to the backup nodes designated in the recovery domain. When an outage occurs on the primary node, the data remains available as a designated backup node takes over as the primary point of access.

"Planning for switchable independent disk pools and cross-site mirroring (XSM)" on page 103
A single copy of the data is maintained on switchable hardware; either an expansion unit (tower) or an IOP in a logical partition environment.

Planning for logical replication:

Multiple copies of the data are maintained with logical replication. Data is replicated, or copied, from the primary node in the cluster to the backup nodes designated in the recovery domain. When an outage occurs on the primary node, the data remains available as a designated backup node takes over as the primary point of access.

Replication makes a copy of something in real time. It is the process of copying objects from one node in a cluster to one or more other nodes in the cluster. Replication makes and keeps the objects on your systems identical. If you make a change to an object on one node in a cluster, this change is replicated to other nodes in the cluster.

You must decide on a software technology to use for logical replication. The following solutions are available for achieving logical replication in your cluster:

- **IBM Business Partners products**

  Data replication software from recognized cluster IBM Business Partners enables you to replicate objects across multiple nodes. See "Cluster middleware IBM Business Partners and available clustering products" on page 91 for details.
- **A custom-written replication application**

  IBM journal management provides a means by which you can record the activity of objects on your system. You can write an application taking advantage of journal management to achieve logical replication. See Journal management for details on how journal management works.

  Related concepts

  Journal management

Determining which systems to use for logical replication:

When you are determining which systems to use for logical replication, there are several key considerations.

These considerations are:

- Performance capacity
- Disk capacity
- Critical data
- Disaster prevention

If your system fails over, you need to know what data and applications you have running on your primary system and your backup system. You want to put the critical data on the system that is most capable of handling the workload in case it fails over. You do not want to run out of disk space. If your primary system runs out of space and fails over, it is highly likely that your backup system is also going to fail over due to lack of disk space. To ensure your data center is not completely destroyed in case of a natural disaster, such as a flood, tornado, or hurricane, you should locate the replicated system in a remote location.

Planning for switchable independent disk pools and cross-site mirroring (XSM):

A single copy of the data is maintained on switchable hardware; either an expansion unit (tower) or an IOP in a logical partition environment.

When an outage occurs on the primary node, access to the data on the switchable hardware switches to a designated backup node. Additionally, independent disk pools can be used in a cross-site mirroring (XSM) environment. This allows a mirror copy of the independent disk pool to be maintained on a system that is (optionally) geographically distant from the originating site for availability or protection purposes.

Careful planning is required if you plan to take advantage of switchable resources residing on switchable independent disk pools or cross-site mirroring (XSM).

Related concepts

Planning for independent disk pools

Cluster security

Consider some of the security issues you need to consider when you plan to use clustering on your systems.

Enabling a node to be added to a cluster

Before you can add a node to a cluster, you need to set a value for the Allow add to cluster (ALWADDCLU) network attribute.

Use the Change Network Attributes (CHGNETA) command on any server that you want to set up as a cluster node. The Change Network Attributes (CHGNETA) command changes the network attributes of a system. The ALWADDCLU network attribute specifies whether a node will allow another system to add it as a node in a cluster.

Note: You must have *IOSYSCFG authority to change the network attribute ALWADDCLU.

You can pick one of these values:

<u>*SAME</u>
The value does not change. The system is shipped with a value of *NONE.

***NONE**
No other system can add this system as a node in a cluster.

***ANY** Any other system can add this system as a node in a cluster.

***RQSAUT**
Any other system can add this system as a node in a cluster only after the cluster add request has been authenticated.

The ALWADDCLU network attribute is checked to see if the node that is being added is allowed to be part of the cluster and whether to validate the cluster request through the use of X.509 digital certificates. A **digital certificate** is a form of personal identification that can be verified electronically. If validation is required, the requesting node and the node that is being added must have the following installed on the systems:
- i5/OS Option 34 (Digital Certificate Manager)
- Cryptographic Access Provider

When *RQSAUT is selected, the certificate authority trust list for the i5/OS cluster security server application must be properly set up. The server application identifier is QIBM_QCST_CLUSTER_SECURITY. At a minimum, add certificate authorities for those nodes that you allow to join the cluster.

Related concepts

Digital Certificate Management

"Common cluster problems" on page 152
Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

Related reference

Change Network Attributes (CHGNETA) command

Distributing cluster-wide information

Learn about the security implications of using and managing cluster-wide information.

The Distribute Information (QcstDistributeInformation) API can be used to send messages from one node in a cluster resource group recovery domain to other nodes in that recovery domain. This can be useful in exit program processing. However, it should be noted that there is no encrypting of that information. Secure information should not be sent using this mechanism unless you are using a secure network.

Non-persistent data can be shared and replicated between cluster nodes using the Clustered Hash Table APIs. The data is stored in non-persistent storage. This means the data is retained only until the cluster node is no longer part of the clustered hash table. These APIs can only be used from a cluster node that is defined in the clustered hash table domain. The cluster node must be active in the cluster.

Other information distributed via cluster messaging is similarly not secured. This includes the low level cluster messaging. As such, when changes are made to the exit program data, there is no encrypting of the message containing that data.

Related reference

Distribute Information (QcstDistributeInformation) API

Clustered Hash Table APIs

Maintaining user profiles on all nodes

You can use two mechanisms for maintaining user profiles on all nodes within a cluster.

| One mechanism is to create a cluster administrative domain to monitor shared resources across nodes in
| a cluster. A cluster administrative domain can monitor several types of resources in addition to user
| profiles, providing easy management of resources that are shared across nodes. See Monitored resources
| for details on these resources. When user profiles are updated, changes are propagated automatically to
| other nodes if the cluster administration domain is active. If the cluster administrative domain is not
| active, the changes will be propagated once the cluster administrative domain is activated.

With the second mechanism, administrators can also use Management Central in iSeries Navigator to perform functions across multiple systems and groups of systems. This support includes some common user-administration tasks that operators need to perform across the multiple systems in their cluster. Management Central allows user profile functions to be performed against groups of systems. The administrator can specify a post-propagation command to be run on the target systems when creating a user profile.

Important:

- If you plan to share user profiles that use password synchronization within a cluster, you must set the Retain Server Security (QRETSVRSEC) system value to 1.
- If you change QRETSVRSEC to 0 after you add an MRE for a user profile, and then change a password (if password is being monitored), the global status for the MRE is set to Inconsistent. The MRE is marked as unusable. Any changes made to the user profile after this change are not synchronized. To recover from this problem, change QRETSVRSEC to 1, remove the MRE, and add the MRE again.

Related concepts

"Job structure and user queues" on page 131
When managing cluster, you need to know about job structures and user queues.

"Cluster administrative domain" on page 9
A *cluster administrative domain* is used to maintain a consistent operational environment across a subset of cluster nodes to ensure that a highly available application will behave as expected as it is switched to or failed over to backup nodes.

Related tasks

"Adding and removing monitored resource entries" on page 122
You can add a monitored resource entry to a cluster administrative domain that represents a resource that is managed by a cluster administrative domain.

Considerations for using clusters with firewalls

If you are using clustering in a network that uses firewalls, you should be aware of some limitations and requirements.

If you are using clustering with a firewall, you need to allow each node the ability to send outbound messages to and receive inbound messages from other cluster nodes. An opening in the firewall must exist for each cluster address on each node to communicate with every cluster address on every other node. IP packets traveling across a network can be of various types of traffic. Clustering uses ping, which is type ICMP, and also uses UDP and TCP. When a user configures a firewall, they can filter traffic based on the type. For clustering to work the firewall needs to allow traffic of ICMP, UDP and TCP. Outbound traffic can be sent on any port and inbound traffic is received on ports 5550 and 5551.

Cluster configuration checklist

Complete the cluster configuration checklist to ensure that your environment is prepared properly before you begin to configure your cluster.

Table 22. TCP/IP configuration checklist for clusters

TCP/IP requirements	
__	Start TCP/IP on every node you plan to include in the cluster using the Start TCP/IP (STRTCP) Command.
__	Configure the TCP loopback address (127.0.0.1) and verify that it shows a status of *Active*. Verify using the Work with TCP/IP Network Status (WRKTCPSTS) Command on every node in the cluster.
__	Verify that the IP addresses used for clustering to a given node must show a status of *Active* using the Work with TCP/IP Network Status (WRKTCPSTS) Command on the subject node.

Table 22. TCP/IP configuration checklist for clusters (continued)

TCP/IP requirements	
__	Verify that INETD is active on all nodes in the cluster by using either the STRTCPSVR *INETD command or through iSeries Navigator by completing the following steps: 1. In iSeries Navigator, expand **Network**. 2. Expand ***your system***. 3. Expand **TCP/IP**. 4. Right-click **INETD** and select **Start**. The INETD server can also be started using the STRTCPSVR (Start TCP/IP Server) command and specifying the *INETD parameter. This can be verified by the presence of a QTOGINTD (User QTCP) job in the Active Jobs list on the subject node.
__	Verify that user profile for INETD, which is specified in /QIBM/ProdData/OS400/INETD/inetd.config, does not have more than minimal authority. The Start Cluster Node (STRCLUNOD) command will fail, if this user profile has more than minimal authority. By default, QUSER is specified as the user profile for INETD.
__	Verify every IP address in the cluster can route to and send UDP datagrams to every other IP address in the cluster. Use the PING command specifying a local IP address and the TRACEROUTE command specifying UDP messages.
__	Verify that ports 5550 and 5551 are not being used by other applications. These ports are reserved for IBM clustering. Port usage can be viewed using the Work with TCP/IP Network Status (WRKTCPSTS) command. Port 5550 will be opened and in a 'Listen' state by clustering once INETD is started.

If you plan to use switchable devices in your cluster, the following requirements must be satisfied:

Table 23. Resilient device configuration checklist for clusters

Resilient device requirements	
__	Verify that Option 41 (HA Switchable Resources) is installed and a valid license key exists on all cluster nodes that will be in the device domain. Note that any use of the iSeries Navigator cluster management interface requires this option.
__	In order to access disk management functions in iSeries Navigator, configure the service tools server (STS) with DST access and user profiles. See Set up communication for details.
__	If you are switching resilient devices between logical partitions on a system, and you are using something other than the HMC to manage your logical partitions, enable Virtual OptiConnect for the partitions. This is done at dedicated service tools (DST) signon. See Virtual OptiConnect for details. If you are using the Hardware Management Console to manage your partitions, change your partition profile properties on the OptiConnect tab to enable Virtual OptiConnect for each partition in the switchable configuration. You must activate the partition profile to reflect the change.
__	If a tower on an HSL OptiConnect loop is switched between two systems, and one of the systems has logical partitions, enable HSL OptiConnect for the partitions. If you are using something other than the HMC to manage logical partitions, this is done at dedicated service tools (DST) signon. If you are using the Hardware Management Console to manage your partitions, change your partition profile properties on the OptiConnect tab to enable HSL OptiConnect for each partition in the switchable configuration. You must activate the partition profile to reflect the change.
__	If you are switching resilient devices between logical partitions, and you are using something other than the HMC to manage your logical partitions, you must configure the bus to be shared between the partitions or configure an I/O Pool. The bus must be configured the bus as ″own bus shared″ by one partition, and all other partitions that will participate in the device switching must be configured as ″use bus shared.″ If you are using the Hardware Management Console to manage your logical partitions, you must configure an I/O Pool that includes the I/O processor, I/O adapter, and all attached resources to allow an independent disk pool to be switchable between partitions. Each partition must have access to the I/O pool. See Make your hardware switchable for more details. For details on physical planning requirements for switchable devices, see Physical planning requirements.

Table 23. Resilient device configuration checklist for clusters (continued)

Resilient device requirements	
__	When switching a tower on a HSL loop between two different systems, the tower must be configured as switchable. See Make your hardware switchable for details.
__	When a tower is added to an existing HSL loop, restart all servers on that same loop.
__	The maximum transmission unit (MTU) for your communication paths must be greater than the cluster communications tuneable parameter, message fragment size. MTU for a cluster IP address can be verified using the Work with TCP/IP Network Status (WRKTCPSTS) command on the subject node. The MTU must also be verified at each step along the entire communications path. It may be easier to lower the message fragment size parameter once the cluster is created than raise the MTU for the communications path. See Tunable cluster communications parameters for more information about message fragment size. You can use the Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo) API to view the current settings of the tuning parameters and the Change Cluster Resource Services (QcstChgClusterResourceServices) API to change the settings.

Table 24. Security configuration checklist for clusters

Security requirements	
__	The ALWADDCLU (Allow Add to Cluster) network attribute must be appropriately set on the target node if trying to start a remote node. This should be set to *ANY or *RQSAUT depending on your environment. If set to *RQSAUT, then i5/OS option 34 (Digital Certificate Manager) and Cryptographic Access Provided Product must be installed. See Enable a node to be added to a cluster for details on setting the ALWADDCLU network attribute.
__	Enable the status of the user profile for INETD specified in /QIBM/ProdData/OS400/INETD/inetd.config. It must not have *SECADM or *ALLOBJ special authorities. By default, QUSER is specified as the user profile for INETD.
__	Verify that the user profile invoking the cluster resource services APIs exists on all cluster nodes and has *IOSYSCFG authority.
__	Verify that the user profile to run the exit program for a cluster resource group (CRG) exists on all recovery domain nodes.

Table 25. Job configuration checklist for clusters

Job considerations	
__	Jobs can be submitted by the cluster resource services APIs to process requests. The jobs will either run under the user profile to run the exit program specified when creating the a cluster resource group, or under the user profile that requested the API (for varying on devices in resilient device CRGs only). The user must ensure that the subsystem which services the job queue associated with the user profile is configured as: *NOMAX for the number of jobs it can run from that job queue.
__	Jobs will be submitted to the job queue specified by the job description which is obtained from the user profile defined for a CRG. The default job description will cause the jobs to be sent to the QBATCH job queue. Since this job queue is used for many user jobs, the exit program job may not run in a timely fashion. Users should consider a unique job description with a unique user queue.
__	When exit program jobs are run, they will use routing data from the job description to choose which main storage pool and run time attributes they will use. The default values will result in jobs that are run in a pool with other batch jobs with a run priority of 50. Neither of these may produce the desired performance for exit program jobs. The subsystem initiating the exit program jobs (the same subsystem that is using the unique job queue) should assign the exit program jobs to a pool that is not used by other jobs initiated by the same subsystem or other subsystems. In addition, the exit program jobs should be assigned a run priority of 15 so that they will run before almost all other user jobs.
__	The QMLTTHDACN system value must be set to 1 or 2.

There are several software solutions available for configuring and managing your cluster. One of these solutions is iSeries Navigator cluster management. If you choose to use iSeries Navigator, the following requirements must be satisfied:

Table 26. iSeries Navigator configuration checklist for clusters

iSeries Navigator cluster management considerations	
__	Option 41 (i5/OS - HA Switchable Resources) must be installed and a valid license key must exist on all cluster nodes that will be in the device domain.
__	Verify that all host servers are started using the STRHOSTSVR (Start Host Server) Command: STRHOSTSVR SERVER(*ALL)
__	Verify that the Management Central server is started using the STRTCPSVR (Start TCP/IP Server) Command: STRTCPSVR SERVER(*MGTC)

Related concepts

"iSeries Navigator cluster management" on page 83
IBM offers a cluster management interface that is available through iSeries Navigator and accessible through Option 41 (i5/OS - HA Switchable Resources).

"Starting the INETD server" on page 94
The Internet daemon (INETD) server must be started in order for a node to be added or started, as well as for merge partition processing.

"Tunable cluster communications parameters" on page 128
The Change Cluster Resource Services (QcstChgClusterResourceServices) API enables some of the cluster topology services and cluster communications performance and configuration parameters to be tuned to better suit the many unique application and networking environments in which clustering occurs. This API is available to any cluster that is running at cluster version 2 or later.

Related reference

"Cluster administrative domain checklist" on page 110
This topic covers all the prerequisites that must be completed before creating a cluster administrative domain.

Cluster deconfiguration checklist

If you need to delete a cluster or a CRG, you must systematically remove different cluster components to ensure complete deconfiguration.

Table 27. Independent disk pool deconfiguration checklist for clusters

Independent disk pool requirements	
__	If you plan to remove a subset of an independent disk pool group or remove the last independent disk pool in the switchable devices, you must end the CRG first. Use the End Cluster Resource Group (ENDCRG) command.
__	If you want delete an independent disk pool that is participating in a cluster, it is strongly recommended that you first remove the configuration object of the disk pool from the switchable device, also known as a device cluster resource group (CRG). To delete configuration object of a disk pool from a switchable device, follow these steps: To delete a disk pool from a switchable device, follow these steps: 1. In iSeries Navigator, expand **Management Central** → **Clusters**. 2. Expand the ***cluster name containing switchable device*** → **Switchable Devices**. 3. Click the name of the switchable device. 4. In the right-hand pane of iSeries Navigator, right-click the disk pool, and select **Remove**. You can also use the Remove CRG Device Entry (RMVCRGDEVE) command to remove the configuration object of the independent disk pool from the CRG.
__	After you have removed the configuration object of the independent disk pool from the cluster switchable device, you can delete an independent disk pool .

Table 27. Independent disk pool deconfiguration checklist for clusters (continued)

Independent disk pool requirements	
__	Delete device description for independent disk pool by completing these tasks: 1. On command line interface, type WRKDEVD DEVD(*ASP) and press Enter. 2. Page down until you see the device description for the independent disk pool that you want to delete. 3. Select Option 4 (Delete) by the name of the device description and press Enter.

Table 28. Cluster resource group deconfiguration checklist for clusters

Cluster resource group requirement	
__	Delete cluster resource group, by completing the either of the following tasks: 1. If clustering is not active on the node, then type DLTCRG CRG(CRGNAME) on a command-line interface. CRGNAME is the name of the CRG that you want to delete. Press Enter. 2. If clustering is active on the node, then type DLTCRGCLU CLUSTER(CLUSTERNAME) CRG(CRGNAME) on a command-line interface. CLUSTERNAME is the name of the cluster. CRGNAME is the name of the CRG that you want to delete. Press Enter.

Planning for a cluster administrative domain

The cluster administrative domain requires planning to manage resources that are synchronized among nodes within a cluster administrative domain. In order to ensure that an application will run consistently on any system in a high availability (HA) environment, all resources that affect the behavior of the application need to be identified, as well as the cluster nodes where the application will run, or where application data might reside.

A cluster administrator can create a cluster administrative domain and add monitored resources that are synchronized among nodes. The i5/OS Cluster provides a list of system resources that can be synchronized by a cluster administrative domain, represented by *monitored resource entries (MREs)*. See Monitored resources for a complete list of system resources that can be monitored.

When designing a cluster administrative domain, you should answer the following questions:

What resources will be synchronized?
You will need to determine which system resources need to be synchronized. You can select attributes for each of these resources to customize what is synchronized. Applications that run on multiple nodes may need specific environment variables to run properly. In addition data that spans several nodes may also require certain user profiles to be accessed. You should be aware of the operational requirements for your applications and data before you determine what resources will be managed by a cluster administrative domain.

What nodes will be included in the cluster administrative domain?
You should determine what nodes in a cluster are to be managed by the cluster administrative domain. These are the cluster nodes representing the systems where an application can run or where the application data is stored, and which require a consistent operational environment. Nodes cannot be in multiple cluster administrative domains. For example, if you have four nodes in a cluster (Node A, Node B, Node C and Node D). Nodes A and B can be in one cluster administrative domain and Nodes C and D can be in another. However you cannot have Node B and C in a third cluster administrative domain and still be in their original cluster administrative domain.

What will be the naming convention for cluster administrative domains?
Depending on the complexity and size of your clustered environment, you may want to establish a standard naming convention for peer CRGs and cluster administrative domains. Since a peer CRG is created when you create a cluster administrative domain, you will want to differentiate other peer CRGs from those that represent cluster administrative domains. For example, peer CRGs that represent cluster administrative domains can be named *ADMDMN1*, *ADMDMN2*, and

so forth, while other peer CRGs can be named *PEER1*. You can also use the List Cluster Resource Group Information (QcstListClusterResourceGroupIn) API to determine whether the peer CRG is used as a cluster administrative domain. A peer CRG which represents a cluster administrative domain can be identified by its application identifier which is QIBM.AdminDomain.

Cluster administrative domain checklist

This topic covers all the prerequisites that must be completed before creating a cluster administrative domain.

Table 29. Cluster administrative domain checklist

Cluster administrative domain requirements	
__	Verify that a cluster has been configured. See Cluster configuration checklist.
__	If you plan to monitor user profiles that use password synchronization within a cluster, you must set the Retain Server Security (QRETSVRSEC) system value to 1.
__	If you change QRETSVRSEC to 0 after you add an MRE for a user profile, and then change a password (if password is being monitored), the global status for the MRE is set to Inconsistent. The MRE is marked as unusable. Any changes made to the user profile after this change are not synchronized. To recover from this problem, change QRETSVRSEC to 1, remove the MRE, and add the MRE again.
__	To add resources to the cluster administrative domain, all the nodes in the cluster administrative domain must be active and participating in the group and not partitioned. See Monitored Resources.

Configuring clusters

Understand how to create a cluster.

IBM and IBM cluster middleware IBM Business Partners have teamed together to provide state-of-the-art cluster resource services functions along with a graphical user interface (GUI) for cluster management. i5/OS cluster resource services provides a set of integrated services that maintain cluster topology, perform heartbeating, and allow creation and administration of cluster configuration and cluster resource groups. Cluster resource services also provides reliable messaging functions that keep track of each node in the cluster and ensures that all nodes have consistent information about the state of cluster resources. In addition, cluster resource services provides a set of control language (CL) commands and application program interfaces (APIs) and facilities that can be used by System i application providers or customers to enhance their application availability. Cluster resource services functions can also be accessed via graphical user interface solutions provided by iSeries Navigator cluster management and cluster middleware IBM Business Partner products.

Getting started

Follow these steps to configure a cluster:

1. **Select a software solution.**

   See “Solutions for configuring and managing clusters” on page 83 for complete look at the options for configuring and managing clusters.
2. **Satisfy the hardware, software and communications requirements.**

   Review the cluster requirements in Plan for clusters.
3. **Set up your network and server environment for clusters.**

   Use the “Cluster configuration checklist” on page 105 to make sure that you are prepared to configure clusters in your environment.
4. **Configure your cluster.**

Related concepts

“Who to call for cluster support” on page 167
See this topic if you need to contact IBM with your cluster questions.

Creating a cluster

To create and configure a cluster, you need to include at least one node in the cluster and you must have access to at least one of the nodes that will be in the cluster.

If only one node is specified, it must be the server that you are currently accessing. For a complete list of requirement for creating clusters, see the "Cluster configuration checklist" on page 105.

If you will be using switchable devices in your cluster, there are additional requirements than that of a cluster that does not use switchable devices. To set up a cluster environment that includes switchable devices, care must be taken so that conflicts are avoided across the cluster. See Create a switchable independent disk pool for step-by-step instructions on how to configure a cluster to use switchable devices.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

iSeries Navigator cluster management features a wizard that takes you through the steps to create and start a simple cluster consisting of one or two cluster nodes. Once you have created a one- or two-node cluster, you can add nodes to it. A cluster created and managed in iSeries Navigator can contain as many as four nodes. This wizard will guide you through the steps to specify servers to include and create cluster resource groups. When you are creating a simple cluster, the server you are creating the cluster on must be one of the nodes.

To create a simple cluster using the New Cluster wizard in iSeries Navigator, follow these steps:

1. In iSeries Navigator, expand **Management Central**.
2. Right-click **Clusters**, and select **New Cluster**.
3. Follow the wizard's instructions to create a cluster.

Once you have created the cluster, be sure to:

1. Add all of the nodes that you want to include in the cluster. Up to four nodes can be added to a cluster created and managed in iSeries Navigator.
2. Add desired nodes to device domains (for use with switchable hardware groups and independent disk pools).
3. Create and start the switchable resources (switchable device, switchable application, and switchable data).

The online help in iSeries Navigator contains step-by-step procedures on completing these tasks.

Using CL commands and APIs

You can also use CL commands or APIs to create a cluster:

1. **Create the cluster.**
   Create Cluster (CRTCLU) command
   Create Cluster (QcstCreateCluster) API
2. **Add nodes to your cluster from the active cluster node.**
   Add Cluster Node Entry (ADDCLUNODE) command
   Add Cluster Node Entry (QcstAddClusterNodeEntry) API
3. **Start a cluster node.**
   Start Cluster Node (STRCLUNOD) command
   Start Cluster Node (QcstStartClusterNode) API
4. **Define device domains.** If you plan to use switchable devices, you must include the desired nodes in the device domain.
   Add Device Domain Entry (ADDDEVDMNE) command

Add Device Domain Entry (QcstAddDeviceDomainEntry) API

5. **Create cluster resource groups (CRG).**
   Create Cluster Resource Group (CRTCRG) command
   Create Cluster Resource Group (QcstCreateClusterResourceGroup) API
6. **Start the cluster resource groups (CRG).**
   Start Cluster Resource Group (STRCRG) command
   Start Cluster Resource Group (QcstStartClusterResourceGroup) API

Managing clusters

This topic contains information that covers some of the tasks that involve managing your clusters.

If you have not considered the type of interface you are going to use to manage your clusters, see Solutions for managing clusters before going any further.

Some of the changes that you can make to the cluster once you have configured it include the following:

Cluster tasks

- Add a node to a cluster
- Remove nodes from a cluster
- Start a cluster node
- End a cluster node
- Adjust the cluster version of a cluster to the latest level
- Delete a cluster
- Change cluster node

Cluster resource group tasks

- Create new cluster resource groups
- Delete existing cluster resource groups
- Start a cluster resource group
- Add a node to a cluster resource group
- Remove a node from a cluster resource group
- End a cluster resource group
- Change the recovery domain for a cluster resource group
- Perform a switchover
- Add a node to a device domain
- Remove a node from a device domain

This topic will also help you to save your cluster configurations. You may want to read about how cluster resource services jobs are structured and how cluster APIs use user queues. Read about the correct way to end cluster jobs and how to monitor cluster status. You also learn how the reliable message function and heartbeat monitoring keep you updated on the status of your cluster.

Cluster administrative domain tasks

- Create a cluster administrative domain
- Add monitored resources
- Delete cluster administrative domain

Related concepts

"Reliable message function" on page 36
The *reliable message function* of cluster resource services keeps track of each node in a cluster and ensures that all nodes have consistent information about the state of cluster resources.

"Heartbeat monitoring" on page 35
Heartbeat monitoring is a cluster resource services function that ensures that each node is active by sending a signal from every node in the cluster to every other node in the cluster to convey that they are still active.

Adding a node to a cluster

You can add a node to a cluster using iSeries Navigator or commands.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

The simple cluster supported by iSeries Navigator can be made up of a maximum of four nodes. If four nodes already exist in the cluster, the **Add Node** option is disabled. If your clustering needs extend beyond four nodes, you should use Cluster commands and APIs or a Cluster middleware IBM Business Partner product to support up to 128 nodes.

To add a node to an existing cluster, follow these steps:

1. In iSeries Navigator, expand Management Central.
2. Expand **Clusters**.
3. Expand the cluster for which you want to add a node.
4. Right-click **Nodes**, and select **Add Node...**

Using Cluster commands and APIs

You can also use the following to add a node to a cluster:

- Add Cluster Node Entry (ADDCLUNODE) command
- Add Cluster Node Entry (QcstAddClusterNodeEntry) API

Related concepts

"Cluster commands and APIs" on page 84
i5/OS cluster resource services provides a set of control language (CL) commands, application program interfaces (APIs) and facilities that can be used by System i application providers or customers to enhance their application availability.

"Cluster middleware IBM Business Partners and available clustering products" on page 91
You can purchase a product from an IBM cluster middleware IBM Business Partner that provides the logical replication functions that are integral to clustering and simplifies cluster creation and management.

Starting a cluster node

Starting a cluster node starts cluster resource services on a node in the cluster. Beginning with cluster version 3, a node can start itself and will be able to rejoin the current active cluster, provided it can find an active node in the cluster.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

When cluster resource services is successfully started on the node specified, the status of the node will be set to *Started*.

To start clustering on a node, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Expand the cluster that contains the node on which you want to start clustering.
4. Click **Nodes**.
5. Right-click the node on which you want to start the cluster, and select **Cluster** → **Start**.

Cluster

Using CL commands and APIs

You can also use CL commands or APIs to start a node. When cluster resource services is successfully started on the node specified, the status of the node will be set to *Active*.
- Start Cluster Node (STRCLUNOD) command
- Start Cluster Node (QcstStartClusterNode) API

Related tasks

"Ending cluster jobs" on page 129
Never try to end a cluster job directly.

"Recovering from cluster job failures" on page 159
Failure of a cluster resource services job is usually indicative of some other problem.

| ## Ending a cluster node

| Stopping or ending a node stops cluster resource services on that node.

| **Using iSeries Navigator**

| This requires Option 41 (HA Switchable Resources) to be installed and licensed.

| When cluster resource services is successfully stopped on the node specified, the status of the node will
| be set to *Stopped*.

| To end clustering on a node, follow these steps:
| 1. In iSeries Navigator, expand **Management Central**.
| 2. Expand **Clusters**.
| 3. Expand the cluster that contains the node on which you want to stop clustering.
| 4. Click **Nodes**.
| 5. Right-click the node on which you want to end clusters, select **Cluster** → **Stop**.

| **Using CL commands and APIs**

| You can also use CL commands or APIs to end a node. When cluster resource services is successfully
| ended on the node specified, the status of the node will be set to *Inactive*.
| - End Cluster Node (ENDCLUNOD) command
| - End Cluster Node (QcstEndClusterNode) API

| **Related tasks**

| "Ending cluster jobs" on page 129
| Never try to end a cluster job directly.

| "Recovering from cluster job failures" on page 159
| Failure of a cluster resource services job is usually indicative of some other problem.

Removing a cluster node

You can remove a node to a cluster using iSeries Navigator or commands.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To remove a node to an existing cluster, follow these steps:

1. In iSeries Navigator, expand Management Central.
2. Expand **Clusters**.
3. Expand the cluster for which you want to remove a node.
4. Right-click **Nodes**, and select **Cluster** → **Remove**

Using Cluster commands and APIs

You can also use the following to remove a node to a cluster:

- Remove Cluster Node Entry (RMVCLUNODE) command
- Remove Cluster Node Entry (QcstRemoveClusterNodeEntry) API

Adjusting the cluster version of a cluster

The cluster version defines the level at which all the nodes in the cluster are actively communicating with each other.

Cluster versioning is a technique that allows the cluster to contain systems at multiple release levels and fully interoperate by determining the communications protocol level to be used.

To change the cluster version, all nodes in the cluster must be at the same potential version. The cluster version can then be changed to match the potential version. This will allow the new function to be used. The version can only be incremented by one. It cannot be decremented without deleting the cluster and recreating it at the lower version. The current cluster version is initially set by the first node defined in the cluster. Subsequent nodes added to the cluster must be equal to the current cluster version or the next level version, otherwise they cannot be added to the cluster.

If you are upgrading a node to a new release, you must ensure that the node has the appropriate cluster version. Clusters only supports a one version difference. If all the nodes in the cluster are at the same release, you should upgrade to the new release, before changing the cluster version. This ensures that all function associated with the new release will be available. See the topic, "Cluster version" on page 20 for detailed actions for upgrading to a new release.

Use the following instructions to both verify and change the cluster version for a node.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To verify or adjust the cluster version of a cluster, follow these steps:

1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Right-click the cluster, and select **Properties**.
4. Verify the cluster version setting or change the version to the desired setting.

Using Cluster commands and APIs

You can also use the following to adjust the cluster version of a cluster:

- Change Cluster Version (CHGCLUVER) command
- Adjust Cluster Version (QcstAdjustClusterVersion) API

Related concepts

"Cluster version" on page 20
A *cluster version* represents the level of function available on the cluster.

"Common cluster problems" on page 152
Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

Related tasks

"Deleting a cluster"
When you delete a cluster, cluster resource services will be ended on all active cluster nodes and they will be removed from the cluster.

Deleting a cluster

When you delete a cluster, cluster resource services will be ended on all active cluster nodes and they will be removed from the cluster.

| **Important:** If you have independent disk pools in your cluster, you should first remove each node from the device domain using the Remove Device Domain Entry (RMVDEVDMNE) command before you delete your cluster.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To delete a cluster, follow these steps:

1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Right-click the cluster you want to delete, and select **Delete**.

Using CL commands and APIs

You can also use CL commands or APIs to delete a cluster.

- | Delete Cluster (DLTCLU) command
- | Delete Cluster (QcstDeleteCluster) API

Related tasks

"Adjusting the cluster version of a cluster" on page 115
The cluster version defines the level at which all the nodes in the cluster are actively communicating with each other.

| Creating a CRG

| You can create several types of CRGs: application, data, device and peer CRGs.

| To create a CRG in a cluster, complete the following steps:

1. | In iSeries Navigator, expand **Management Central** → **Clusters**.
2. | Expand the cluster where you want to add the CRG.
   a. | If you want to create a Device CRG, right-click **Switchable Hardware**, and select **New Group**. Note: The **New Group** option is only available if all the nodes in the recovery domain are started. See start a cluster node for details.

   b. If you want to create an Application CRG, right-click **Switchable Software**, and select **Add Product**.
   c. If you want to create a Data CRG, right-click **Switchable Data**, and select **New Group** .
   d. If you want to create a Peer CRG, right-click **Peer Resources**, and select **New Peer CRG**.

Using CL commands and APIs

You can use the following commands and APIs to create a CRG:
- Create Cluster Resource Group (CRTCRG) command
- Create Cluster Resource Group (QcstCreateClusterResourceGroup) API

Creating an application CRG with an active takeover IP address

You can specify to allow an active takeover IP address when you create an application CRG. This is only allowed if the user configures the takeover IP address.

Previously you could create an application CRG with an active takeover IP address provided it was configured by the user. But the application CRG could not be started if the takeover IP address was already active. Now, you can specify to allow an active takeover IP address when you create the application CRG. When you start an application CRG that allows for an active takeover IP addresses, the CRG will be allowed to start.

To allow for an active takeover IP address when you create an application CRG, complete the following steps:
1. In a command line interface, type:
   ```
   CRTCRG CLUSTER(MYCLUSTER) CRG(MYCRG) CRGTYPE(*APP) EXITPGM(QDEVELOP/EXITPGM)
   USRPRF(USER) RCYDMN((NODE1 *PRIMARY)(NODE2 *BACKUP)) TKVINTNETA('10.1.2.1') CFGINTNETA(*USR *YES)
   ```
   The **TKVINTNETA** parameter identifies the takeover IP address to be used, and the **CFGINTNETA** parameter identifies that the user will configure the takeover IP address and that this address can be active at Start CRG time.

After you have created the application CRG to allow an active takeover IP address, you can start the CRG.

Starting a CRG

You can start several types of CRGs: application, data, device and peer CRGs.

To start a CRG, complete the following tasks:
1. In iSeries Navigator, expand **Management Central** → **Clusters**.
2. Expand the cluster where you want to start the CRG.
   a. If you want to start a Device CRG, click **Switchable Hardware**, right-click the switchable hardware group you would like to start, and select **Start**.
   b. If you want to start an Application CRG, click **Switchable Software**, right-click the switchable software product you would like to start, and select **Start**.
   c. If you want to start a Data CRG, click **Switchable Data**, right-click the switchable data group you want to start, and select **Start**.
   d. If you want to start a Peer CRG, click **Peer Resources** to list all the peer CRGs, right-click the peer CRG you want to start, and select **Start**.

Using CL commands and APIs

You can use the following commands and APIs to start a CRG:
- Start Cluster Resource Group (STRCRG) command

| • Start Cluster Resource Group (QcstStartClusterResourceGroup) API

Changing the recovery domain for a cluster resource group

You can change the roles of nodes in a recovery domain for a cluster resource group, as well as add or remove nodes from a recovery domain. For a device cluster resource group, you can also change site name and data port IP addresses for a node in the recovery domain.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To change the role of nodes in the recovery domain for a cluster resource group (switchable hardware, switchable software, or switchable data), or to add or remove nodes to the recovery domain, follow these steps:

1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Expand the cluster containing the switchable hardware, software, or data for which you want to change the recovery domain.
4. Expand the switchable hardware, software, or data.
5. Right-click the switchable hardware, software, or data, and select **Properties**.
6. Select the **Recovery Domain** page.

Click Help on the Recovery Domain page for instructions on how to change roles or add or remove nodes.

Using CL commands and APIs

To change the role of nodes in the recovery domain, or add or remove nodes, use the following CL commands and APIs:

- Add Cluster Resource Group Node Entry (ADDCRGNODE) command
- Add a Node to Recovery Domain (QcstAddNodeToRcvyDomain) API
- | Change Cluster Resource Group (CHGCRG) command
- | Change Cluster Resource Group (QcstChangeClusterResourceGroup) API
- Remove Cluster Resource Group Node Entry (RMVCRGNODE) command
- Remove Node from Recovery Domain) QcstRemoveNodeFromRcvyDomain API

Related concepts

"Recovery domain" on page 18

A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

Performing a switchover

Performing a manual switchover causes the current primary node to switch over to the backup node, as defined in the cluster resource group's recovery domain.

When this happens, the current roles of the nodes in the recovery domain of a cluster resource group change such that:

- The current primary node is assigned the role of last active backup.
- The current first backup is assigned the role of primary.
- Subsequent backups are moved up one in the order of backups.

| A switchover is only allowed on primary-backup model CRGs that have a status of ACTIVE.

Note: If you are performing a switchover on a switchable device (also known as a device CRG), you should synchronize user profile name, UID, and GID for performance reasons.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To switch a resource - a switchable hardware group, switchable application, or switchable data group - from the primary node to the backup node in the recovery domain, the resource must have a status of **Started**.

To perform a switchover on a resource, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Expand the cluster containing the desired resource.
4. Click **Switchable Hardware**, **Switchable Software**, or **Switchable Data**.
5. Right-click the desired resource, and select **Switch**.

Using Cluster APIs

You can also use the following to perform a switchover:
- Change Cluster Resource Group Primary (CHGCRGPRI) command
- Initiate Switchover (QcstInitiateSwitchOver) API

Related concepts

"Recovery domain" on page 18
A *recovery domain* is a subset of cluster nodes that are grouped together in a cluster resource group (CRG) for a common purpose such as performing a recovery action or synchronizing events.

Related tasks

"Switchover" on page 29
Switchover happens when you manually switch access to a resource from one server to another.

Synchronize user profile name, UID, and GID

Adding a node to a device domain

A device domain is a subset of nodes in a cluster that share device resources.

Before a node can be added to the recovery domain for a device cluster resource group (CRG), the node must be first defined as a member of a device domain. All nodes that will be in the recovery domain for a device CRG must be in the same device domain. A cluster node can belong to at most one device domain.

To create and manage device domains, you must have Option 41 (HA Switchable Resources) installed and a valid license key must exist on all cluster nodes that will be in the device domain.

Using iSeries Navigator

To add a node to a device domain in iSeries Navigator, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Expand the cluster that contains the node you want to add to the device domain.
4. Click **Nodes**.
5. Right-click the node you want to add to the device domain, and select **Properties**.

6. On the **Clustering** page, specify the name of the device domain you want to add the node to in the **Device domain** field.

Using CL commands and APIs

You can also use the following to add a node to a device domain:
- Add Device Domain Entry (ADDDEVDMNE) command
- Add Device Domain Entry (QcstAddDeviceDomainEntry) API

  Related concepts

  "Device domains" on page 25
  A *device domain* is a subset of nodes in a cluster that share device resources. More specifically, nodes in a device domain can participate in a switching action for some collection of resilient device resources.

  Related tasks

  "Removing a node from a device domain"
  A device domain is a subset of nodes in a cluster that share device resources.

Removing a node from a device domain

A device domain is a subset of nodes in a cluster that share device resources.

Important:

Be cautious when removing a node from a device domain. If you remove a node from a device domain, and that node is the current primary point of access for any independent disk pools, those independent disk pools remain with the node being removed. This means that those independent disk pools will no longer be accessible from the remaining nodes in the device domain.

Once a node is removed from a device domain, it cannot be added back to the same device domain if one or more of the existing cluster nodes still belong to that same device domain. In order to add the node back to the device domain, you must:
1. Delete the independent disk pools currently owned by the node being added to the device domain.
2. Perform a system restart (IPL) on the node.
3. Add the node to the device domain. See Add a node to a device domain.
4. Recreate the independent disk pools deleted in Step 1.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To remove a node from a device domain in iSeries Navigator, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Expand the cluster that contains the node you want to remove from the device domain.
4. Click **Nodes**.
5. Right-click the node you want to remove from the device domain, and select **Properties**.
6. On the Clustering page, remove the entry in the **Device domain** field.

Using CL commands and APIs

You can also use the following to remove a node from a device domain:

- Remove Device Domain Entry (RMVDEVDMNE) command
- Remove Device Domain Entry (QcstRemoveDeviceDomainEntry) API

Related concepts

"Device domains" on page 25
A *device domain* is a subset of nodes in a cluster that share device resources. More specifically, nodes in a device domain can participate in a switching action for some collection of resilient device resources.

Related tasks

"Adding a node to a device domain" on page 119
A device domain is a subset of nodes in a cluster that share device resources.

Adding a disk unit or disk pool

How a system event affects a cluster

Certain commands that end system functions, such as Power Down System (PWRDWNSYS), End System (ENDSYS), and End Subsystem (ENDSBS) commands may end a cluster abruptly, causing a cluster partition to occur.

In i5/OS V5R4, enhancements have been made to the PWRDWNSYS, ENDSYS, and ENDSBS commands. If clustering is active on the node when these commands are run, the End Cluster Node (QcstEndClusterNode) API is run.

If you want these commands to complete, you should use OPTION(*CNTRLD) and specify the appropriate delay time in the DELAY parameter. Otherwise, the End Cluster Node API may not have completed before control is returned to the end system function.

Note: If the user specifies **OPTION(*IMMED)**, the End Cluster Node (QcstEndClusterNode) API has about 30 seconds to complete before the system ends. This might result in a failover instead of an end cluster node.

Creating a cluster administrative domain

A cluster administrative domain can be created by using iSeries Navigator or the Create Cluster Administrative Domain (CRTADMDMN) command.

To create the cluster administrative domain, a user must have *IOSYSCFG authority and authority to the QCLUSTER user profile. To manage a cluster administrative domain, a user must be authorized to the CRG that represents the cluster administrative domain, the QCLUSTER user profile, and CRG commands.

Using iSeries Navigator

To create a cluster administrative domain, follow these steps:
1. In iSeries Navigator, expand **Management Central** → **Clusters**.
2. Expand the cluster for which you want to add the cluster administrative domain.
3. Right-click **Peer Resources**, and select **New Administrative Domain**.

Using CL commands and APIs

You can use the following commands and APIs to create a cluster administrative domain:
- Create Cluster Administrative Domain (CRTADMDMN) command
- There is no API that creates a cluster administrative domain.

Related concepts

"Resilient environments" on page 24
Applications running in a cluster need consistent environments to operate properly. Environment resiliency allows changes to objects and attributes to be automatically propagated throughout a cluster.

"Cluster administrative domain" on page 9
A *cluster administrative domain* is used to maintain a consistent operational environment across a subset of cluster nodes to ensure that a highly available application will behave as expected as it is switched to or failed over to backup nodes.

"Monitored resources" on page 10
Monitored resources are types of system resources that can be managed by a cluster administrative domain. These resources are represented in the cluster administrative domain as *monitored resource entries (MREs)*.

"Cluster commands and APIs" on page 84
i5/OS cluster resource services provides a set of control language (CL) commands, application program interfaces (APIs) and facilities that can be used by System i application providers or customers to enhance their application availability.

Adding and removing monitored resource entries

You can add a monitored resource entry to a cluster administrative domain that represents a resource that is managed by a cluster administrative domain.

To add a monitored resource entry, follow these steps:
1. In iSeries Navigator, expand **Management Central** → **Clusters**.
2. Expand the cluster where you want to add the monitored resource entry.
3. Expand **Peer Resources** to view a list of all peer resources in the cluster.
4. Expand the cluster administrative domain where you want to add the monitored resource entry.
5. Right-click a monitored resource type and select **Add Monitored Resource Entry**.
6. Select the attributes to be monitored for the monitored resource entry, and click **OK**.

Restriction:
- IBM-supplied user profiles (any user profile with a name beginning with "Q") cannot be monitored.
- The IBM-supplied classes QARBCLS, QMONCLS, and QLPINSTALL cannot be monitored.

You can remove a monitored resource entry from a cluster administrative domain if you do not wish to have changes synchronized for the resource. Removing the MRE does not delete the resource from the system. To remove a monitored resource entry, follow these steps:
1. In iSeries Navigator, expand **Management Central** → **Clusters**.
2. Expand the cluster from which you want to remove the MRE.
3. Expand **Peer Resources** to view a list of all peer resources in the cluster.
4. Expand the type of MRE that you are interested in.
5. Right-click the MRE that you want to remove and select **Remove Monitored Resource Entry**.

Using CL commands and APIs

You can use the following commands and APIs to add and remove monitored resources:
- There are no supported CL command equivalents for these functions. The source for non-supported commands and command processing programs (CPPs) have been provided in the QUSRTOOL library. To learn about the command source and CPPs, look at the member TFPADINFO in file QATTINFO.
- Add Monitored Resource Entry (QfpadAddMonitoredResourceEntry) API

- Remove Monitored Resource Entry (QfpadRmvMonitoredResourceEntry) API

Related concepts

"Maintaining user profiles on all nodes" on page 104
You can use two mechanisms for maintaining user profiles on all nodes within a cluster.

Managing a cluster administrative domain

Once a cluster administrative domain is created and the appropriate monitored resource entries (MREs) are added, the cluster administrator should monitor the activity within the administrative domain to ensure that the monitored resources remain consistent. iSeries Navigator provides the user interface to monitor a cluster administrative domain. It provides the ability to list the MREs along with the global status for each resource. Detailed information can be displayed by selecting an MRE. This information includes the global value for each attribute associated with the MRE, along with an indication whether the attribute is consistent or inconsistent with the domain.

If the global status of a monitored resource is inconsistent, the administrator should take the necessary steps to determine why the resource is inconsistent, correct the problem and resynchronize the resource.

If the resource is inconsistent because an update failed on one or more nodes, there is information kept for the MRE that will help you determine the cause of the failure. On the node where the failure occurred, a message is logged internally with the MRE which is the cause of the failed update. On the other nodes, there will be an informational message logged internally which tells you there was a failure, along with the list of nodes where the update failed. These messages are available through iSeries Navigator or by calling the Retrieve Monitored Resource Information (QfpadRtvMonitoredResourceInfo) API. Failure messages are also logged in the joblog of the peer CRG job.

Once the cause of the inconsistency is determined, the resource can be resynchronized, either as a result of an update operation on the node where the failure occurred, or by ending and restarting the administrative domain. For example, an MRE for a user profile could be inconsistent because you changed the UID for the user profile on one node in the administrative domain, but the UID you specified was already in use by another user profile on one of the nodes. If you change the value of the UID again to something that is not used by another user profile within the administrative domain, the change will be made by the cluster administrative domain on all nodes and the global status for the user profile MRE will be set to consistent. You do not need to take any further action to resynchronize the user profile MRE.

In some cases, you will need to end and restart the cluster administrative domain CRG in order for the inconsistent resources to be resynchronized. For example, if you change the UID for a user profile that has an MRE associated with it, but the user profile is active in a job on one of the other cluster nodes in the administrative domain, the global value for the MRE associated with the user profile will be set to *inconsistent* because the change operation failed on the node where the user profile was active in a job. In order to correct this situation, you need to wait until the job has ended and then end the cluster administrative domain. When the administrative domain is started again, the global value for each attribute that is inconsistent will be used to change the resource to a consistent state.

The global status for a monitored resource is always set to *inconsistent* if the resource is deleted, renamed or moved on any node in the domain. If this is the case, the MRE should be removed because the resource will no longer be synchronized by the cluster administrative domain.

When you restore a monitored resource on any system which is part of a cluster administrative domain, the resource will be resynchronized to the global value currently known in the cluster administrative domain when the peer CRG representing the cluster administrative domain is active.

The following restore commands will result in a resynchronization of system objects: RSTLIB, RSTOBJ, RSTUSRPRF and RSTCFG. In addition, RSTSYSINF and UPDSYSINF will result in a resynchronization of system values and network attributes. To resynchronize system environment variables after RSTSYSINF or UPDSYSINF, the peer CRG that represents the cluster administrative domain must be ended and started again.

If you want to restore your monitored resources to a previous state, remove the MRE that represents the resource that you want to restore. Then, after restoring the resource, add an MRE for the resource from the system where the restore operation was done. The cluster administrative domain will synchronize the monitored resource across the domain using the values from the restored resource.

There are considerations for the following actions when using a cluster administrative domain:
- Adding a node to the cluster administrative domain
- Starting and ending the peer CRG
- Starting and ending a cluster node
- Partitioned cluster administrative domains

Using iSeries Navigator

To monitor a cluster administrative domain, follow these steps:
1. In iSeries Navigator, expand **Management Central** → **Clusters**.
2. Expand the cluster for which the cluster administrative domain is associated.
3. Expand **Peer Resources**.
4. Expand the cluster administrative domain that you are interested in.
5. Expand the type of monitored resource you are interested in and click the resource. The possible values for the status of the resource across the active cluster administrative domain are:

   Consistent
   The values for all the resource's attributes monitored by the system are the same on all active nodes within the cluster administrative domain.

   Inconsistent
   The values for all the resource's attributes monitored by the system are not the same on all active nodes within the cluster administrative domain, or the cluster administrative domain is not active.

   Pending
   The values of the monitored attributes are in the process of being synchronized across the cluster administrative domain.

   Added
   The monitored resource entry has been added to the monitored resource directory in the cluster administrative domain but has not yet been synchronized.

Using CL commands and APIs

You can use the following commands and APIs to monitor a cluster administrative domain:
- There is no supported CL command equivalent for this function. The source for a non-supported command and call processing program (CPP) has been provided in the QUSRTOOL library. To learn about this command source and CPP, look at the member TFPADINFO in file QATTINFO.
- Retrieve Monitored Resource Information (QfpadRtvMonitoredResourceInfo) API

Adding a node to the cluster administrative domain

Consider the following information when adding a node to the cluster administrative domain.

You can add additional cluster nodes to your cluster administrative domain by adding a node to the peer CRG which represents the cluster administrative domain. See Add CRG Node Entry (ADDCRGNODE) for instructions. When you add a node to the administrative domain, the monitored resource entries (MREs) from the domain are copied to the node being added. If the monitored resource does not exist on the new node, it is created by the cluster administrative domain. If the monitored resource already exists on the node being added, it is synchronized with the rest of the cluster administrative domain if the domain is active. That is, the values of the attributes for each monitored resource on the node that is joining are changed to match the global values for the monitored resources in the active domain.

Starting and ending the peer CRG

There are several considerations to recognize when starting and ending a peer cluster resource group (CRG).

A cluster administrative domain is inactive when the peer CRG is ended. While the cluster administrative domain is inactive, all of the monitored resources are considered to be inconsistent because changes to them are not being synchronized. Although changes to monitored resources continue to be tracked, the global value is not changed and changes are not propagated to the rest of the administrative domain. Any changes made to any monitored resource while the cluster administrative domain is inactive are synchronized across all active nodes when the peer CRG is started.

During the synchronization process, the cluster administration attempts to change each resource with attributes whose values do not match its global values, unless there is a pending change for that resource. Any pending change is distributed to all active nodes in the domain and applied to each affected resource on each node. When the pending changes are distributed, the global value is changed and the global status of each affected resource is changed to *consistent* or *inconsistent*, depending on the outcome of the change operation for the resource on each node. If the affected resource is changed successfully on every active node in the domain, the global status for that resource is *consistent*. If the change operation failed on any node, the global status is set to *inconsistent*.

If changes are made to the same resource from multiple nodes while the cluster administrative domain is inactive, all of the changes are propagated to all of the active nodes as part of the synchronization process when the CRG is started. Although all pending changes are processed during the activation of the cluster administrative domain, there is no guaranteed order in which the changes are processed. If you make changes to a single resource from multiple cluster nodes while the CRG is inactive, there is no guaranteed order to the processing of the changes during activation.

If a node joins an inactive admin domain (that is, the node is started while the cluster administrative domain is ended), the monitored resources are not resynchronized until the peer CRG is started.

When the cluster administrative domain peer CRG is started, any change made to any monitored resource while the cluster administrative domain was ended will be propagated to all active nodes in the administrative domain.

Important: The cluster administrative domain peer CRG and its associated exit program are IBM-supplied objects. They should not be changed using the QcstChangeClusterResourceGroup API or the CHGCRG command. Making these changes will cause unpredictable results.

Starting or ending a cluster administrative domain node

Consider the following information when starting and ending a cluster node.

After a cluster node that is part of a cluster administrative domain is ended, monitored resources can still be changed on the inactive node. When the node is started again, the changes will be resynchronized with the rest of the administrative domain. During the resynchronization process, the cluster administrative domain applies any changes from the node that was inactive to the rest of the active nodes in the domain, unless changes had also been made in the active domain while the node was inactive. If changes were made to a monitored resource both in the active domain and on the inactive node, the changes made in the active domain are applied to the joining node. In other words, no changes made to any monitored resource are lost, regardless of the status of the node.

If you want to end a cluster node that is part of a cluster administrative domain, and not allow changes made on the inactive node to be propagated back to the active domain when the node is started (for example, when ending the cluster node to do testing on it), you must remove the node from the administrative domain peer CRG before you end the cluster node. See Remove CRG Node Entry (RMVCRGNODE) for details.

Related reference

Remove CRG Node Entry (RMVCRGNODE) command

Partitioned cluster administrative domains

Consider the following information when working with partitioned cluster administrative domains.

If a cluster administrative domain is partitioned, changes continue to be synchronized among the active nodes in each partition. When the nodes are merged back together again, the cluster administrative domain propagates all changes made in every partition so that the resources are consistent within the active domain. There are several considerations regarding the merge processing for a cluster administrative domain:

- If all partitions were active and changes were made to the same resource in different partitions, the most recent change is applied to resource on all nodes during the merge. The most recent change is determined using Coordinated Universal Time (UTC) from each node where a change initiated.
- If all partitions were inactive, the global values for each resource are resolved based on the last change made while any partition was active. The actual application of these changes to the monitored resources does not happen until the peer CRG that represents the cluster administrative domain is started.
- If some partitions were active and some were inactive prior to the merge, the global values representing changes made in the active partitions are propagated to the inactive partitions. The inactive partitions are then started, causing any pending changes made on the nodes in the inactive partitions to propagate to the merged domain.

Monitoring cluster status

Taking appropriate actions when necessary, the cluster resource services performs basic monitoring of a cluster and its components using the reliable message function and heartbeat monitoring.

You can also manually monitor the status of a cluster and its components.

Using iSeries Navigator

This requires Option 41 (HA Switchable Resources) to be installed and licensed.

To monitor the status of a cluster in iSeries Navigator:

1. In iSeries Navigator, expand Management Central.
2. Expand **Clusters**.
3. Navigate within the iSeries Navigator folders for the desired cluster to view the status of the cluster, its nodes, and resources using the Status column in the iSeries Navigator list. The online help contains descriptions of the possible values for the Status column. You can also right-click on components of the cluster and select **Properties** to view information about the cluster.

Using CL commands and APIs

You can use the following commands and APIs to monitor cluster status:

Cluster Information

Retrieves information about a cluster, such as the nodes in the cluster, which adapter IP addresses are being used on each node, and the status of each node in the cluster.

- Display Cluster Information (DSPCLUINF) command
- List Cluster Information (QcstListClusterInfo) API
- List Device Domain Info (QcstListDeviceDomainInfo) API
- Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo) API
- Retrieve Cluster Information (QcstRetrieveClusterInfo) API

Cluster Resource Group information

Generates a list of cluster resource groups and information about the cluster resource group in the cluster, such as the name of the primary node of each CRG in the cluster.

- Display Cluster Resource Group Information (DSPCRGINF) command
- List Cluster Resource Groups (QcstListClusterResourceGroups) API
- List Cluster Resource Group Information (QcstListClusterResourceGroupInf) API

Related concepts

"Reliable message function" on page 36
The *reliable message function* of cluster resource services keeps track of each node in a cluster and ensures that all nodes have consistent information about the state of cluster resources.

"Heartbeat monitoring" on page 35
Heartbeat monitoring is a cluster resource services function that ensures that each node is active by sending a signal from every node in the cluster to every other node in the cluster to convey that they are still active.

Cluster performance

When changes are made to a cluster, the overhead necessary to manage the cluster can be affected.

The only resources that clustering requires are those necessary to perform heartbeat monitoring, to manage the cluster resource groups and the cluster nodes, and to handle any messaging taking place between cluster resource groups and cluster nodes. Once your clustering environment is operational, the only increase in overhead will be if you make changes to the cluster.

During a normal operating environment, there should be minimal impact to your clustered systems due to clustering activity.

Related concepts

"Heartbeat monitoring" on page 35
Heartbeat monitoring is a cluster resource services function that ensures that each node is active by sending a signal from every node in the cluster to every other node in the cluster to convey that they are still active.

"Common cluster problems" on page 152
Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

Balancing network load for clusters

Balance the network load by splitting your work between the communication lines that you use to connect the nodes in the cluster.

The more you can balance the work to keep resource utilization low, the more smoothly your system will run.

CPU load on backup nodes:

Utilize your backup systems as much as possible, but you need to be aware that additional workload can be transferred over to the backup node if a failover should occur.

It is very important to be aware of what is critical to your business and what is not. If you have highly critical applications that are failing over, you need to ensure that the central processing unit (CPU) load on your backup nodes is not so high that the critical applications cannot run.

Tuning cluster performance

Since potentially significant differences exist in your communications environment, you have the capability to adjust variables that affect cluster communications to best match your environment.

The default values should normally be acceptable to most common environments. If your particular environment is not well suited for these defaults, you can tune cluster communications to better match your environment. Two levels of tuning are available.

Base-level tuning

Base-level tuning allows you to set the tuning parameters to a predefined set of values identified for high, low, and normal timeout and messaging interval values. When the normal level is selected, the default values are used for cluster communications performance and configuration parameters. Selecting the low level causes clustering to increase the heartbeating interval and the various message timeout values. With fewer heartbeats and longer timeout values, the cluster will be less sensitive to communications failures. Selecting the high level causes clustering to decrease the heartbeating interval and the various message timeout values. With more frequent heartbeats and shorter timeout values, the cluster will be more sensitive to communications failures.

Advanced-level tuning

Advanced-level tuning is also available such that individual parameters may be tuned over predefined ranges of values. This allows more granular tuning to meet any special circumstances in your communications environment. If an advanced level of tuning is desired, it is recommended that you obtain help from IBM support personnel or equivalent. Setting the individual parameters incorrectly can easily result in decreased performance.

Related reference

Change Cluster Resource Services (QcstChgClusterResourceServices) API

Tunable cluster communications parameters:

The Change Cluster Resource Services (QcstChgClusterResourceServices) API enables some of the cluster topology services and cluster communications performance and configuration parameters to be tuned to better suit the many unique application and networking environments in which clustering occurs. This API is available to any cluster that is running at cluster version 2 or later.

The Change Cluster Configuration Tuning (CHGCLUCFG) command provides a base level of tuning, while the QcstChgClusterResourceServices API provides both base and advanced levels of tuning.

The QcstChgClusterResourceServices API and CHGCLUCFG command can be used to tune cluster performance and configuration. The API and command provide a base level of tuning support where the cluster will adjust to a predefined set of values identified for high, low, and normal timeout and messaging interval values. If an advanced level of tuning is desired, usually anticipated with the help of IBM support personnel, then individual parameters may be tuned through use of the API over a predefined value range. Inappropriate changes to the individual parameters can easily lead to degraded cluster performance.

When and how to tune cluster parameters?

The CHGCLUCFG command and the QcstChgClusterResourceServices API provide for a fast path to setting cluster performance and configuration parameters without your needing to understand details. This base level of tuning primarily affects the heartbeating sensitivity and the cluster message timeout values. The valid values for the base level of tuning support are:

1 (High Timeout Values/Less Frequent Heartbeats)

2 (Default Values)
Normal default values are used for cluster communications performance and configuration parameters. This setting may be used to return all parameters to the original default values.

3 (Low Timeout Values / More Frequent Heartbeats)
Adjustments are made to cluster communications to decrease the heartbeating interval and decrease the various message timeout values. With more frequent heartbeats and shorter timeout values, the cluster is quicker to respond (more sensitive) to communications failures.

Resultant example response times are shown in the following table for a heartbeat failure leading to a node partition:

	1 (Less sensitive)			**2 (Default)**			**3 (More sensitive)**		
	Detection of Heartbeat Problem	Analysis	Total	Detection of Heartbeat Problem	Analysis	Total	Detection of Heartbeat Problem	Analysis	Total
Single subnet	00:24	01:02	01:26	00:12	00:30	00:42	00:04	00:14	00:18
Multiple subnets	00:24	08:30	08:54	00:12	04:14	04:26	00:04	02:02	02:06

| **Note:** Times are specified in minutes:seconds format.

Depending on typical network loads and specific physical media being used, a cluster administrator might choose to adjust the heartbeating sensitivity and message timeout levels. For example, with a high speed high-reliability transport, such as OptiConnect with all systems in the cluster on a common OptiConnect bus, one might desire to establish a more sensitive environment to ensure quick detection leading to faster failover. Option 3 is chosen. If one were running on a heavily loaded 10Mbs Ethernet bus and the default settings were leading to occasional partitions just due to network peak loads, option 1 could be chosen to reduce clustering sensitivity to the peak loads.

The Change Cluster Resource Services API also allows for tuning of specific individual parameters where the network environmental requirements present unique situations. For example, consider again a cluster with all nodes common on an OptiConnect bus. Performance of cluster messages can be greatly enhanced by setting the Message Fragment Size parameter to the maximum 32,500 bytes to better match the OptiConnect Maximum Transmission Unit (MTU) size than does the default 1,464 bytes. This reduces the overhead of fragmentation and reassembly of large messages. The benefit, of course, depends on the cluster applications and usage of cluster messaging resulting from those applications. Other parameters are defined in the API documentation and may be used to tune either the performance of cluster messaging or change the sensitivity of the cluster to partitioning.

Ending cluster jobs

Never try to end a cluster job directly.

If you need to stop whatever may be running in a clustered environment, you should:

1. End the cluster node.
2. Fix the problem.
3. Start the cluster node.

Related tasks

"Ending a cluster node" on page 114
Stopping or ending a node stops cluster resource services on that node.

"Starting a cluster node" on page 113
Starting a cluster node starts cluster resource services on a node in the cluster. Beginning with cluster version 3, a node can start itself and will be able to rejoin the current active cluster, provided it can find an active node in the cluster.

Resource Monitoring and Control (RMC)

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

RMC is utilized as a communication mechanism for reporting service events to the Hardware Management Console (HMC). If RMC is not active, then service events will not be reported to the HMC. The following list describes services that are associated with RMC:

CAS Daemon

Purpose: Acts as the authentication server for RMC.

Job Name: QCSTCTCASD

RMC Daemon

Purpose: Monitors of resources by communicating with the Resources Managers.

Job Name: QCSTCTRMCD

SRC Daemon

Purpose: Monitors the status of the other RMC jobs; it will restart a job if that particular job unexpectedly ends.

Job Name: QCSTSRCD

Resource Managers (RM)

A Resource Manager (RM) is a job that manages and provides the interface between RMC and actual physical or logical entities. Although RMC provides the basic abstractions, such as resource classes, resources, and attributes for representing physical or logical entities, it does not itself represent any actual entities. An RM maps actual entities to RMC's abstractions. The following list describes the different Resource Managers that are supported for RMC:

Audit Log RM

Purpose: Provides a facility for recording information about the system's operation.

Job Name: QYUSALRMD

CSMAgent RM

Purpose: Provides resource classes to represent the Management Server, which is the HMC.

Job Name: QYUSCMCRMD

Host RM

Purpose: Provides resource classes to represent an individual machine.

Job Name: QCSTCTHRMD

Service RM

Purpose: Manages problem information and prepares it for delivery to the HMC.

Job Name: QSVRMSERMD

Starting or ending the RMC

All RMC jobs, including RM jobs are in the QSYSWRK subsystem and are automatically started when the subsystem is started. TCP/IP must be active for startup to complete. The RMC Daemon requires TCP/IP to be active. If TCP/IP becomes inactive, then the RMC Daemon will end. The RMC Daemon will be automatically restarted by the SRC Daemon once TCP/IP becomes active again. No steps are required of the user under normal conditions. If RMC needs to be manually started, run the following command:

```
SBMJOB CMD(CALL PGM(QSYS/QCSTCTSRCD)) JOBD(QSYS/QCSTSRCD) PRTDEV(*JOBD) OUTQ(*JOBD)
USER(*JOBD) PRTTXT(*JOBD) RTGDTA(RUNPTY50)
```

If RMC needs to be manually ended, use the ENDJOB command to end the QCSTSRCD job. This command should end all RMC jobs. If all the jobs do not end, then manually end each of the jobs listed above.

Job structure and user queues

When managing cluster, you need to know about job structures and user queues.

Cluster resource services job structure

Cluster resource services consists of a set of multi-threaded jobs. When clustering is active on a server, the following jobs run in the QSYSWRK subsystem under the QSYS user profile. The jobs run using the QDFTSVR job description, but with the logging level set such that a job log will be produced.

- Cluster control consists of one job that is named QCSTCTL.
- Cluster resource group manager consists of one job that is named QCSTCRGM.
- Cluster resource groups consist of one job per cluster resource group object. The job name is the same as the cluster resource group name. This includes the cluster administrative domain.
- When one or more device list entries in a resilient device CRG have been set to be brought online at switchover or failover, additional jobs will be submitted to perform the vary on function.

The QCSTCTL and QCSTCRGM job are cluster critical jobs. That is, the jobs must be running in order for the node to be active in the cluster.

Most cluster resource group APIs result in a separate job being submitted that use the user profile specified when the cluster resource group was created. The exit program defined in the cluster resource group is called in the submitted job. By default, the jobs are submitted to the QBATCH job queue. Generally, this job queue is used for production batch jobs and will delay or prevent completion of the exit programs. To allow the APIs to run effectively, create a separate user profile, job description, and job queue for use by cluster resource groups. Specify the new user profile for all cluster resource groups that you create. The same program is processed on all nodes within the recovery domain that is defined for the cluster resource group.

You can use the Change Cluster Recovery (CHGCLURCY) command to restart the cluster resource group job that ended without ending and restarting clustering on a node.

Cluster APIs use of user queues

Functions performed by an API that has a results information parameter operate asynchronously and send their results to a user queue once the API is finished processing. The user queue must be created before calling the API. You can create a user queue by using the Create User Queue (QUSCRTUQ) API. The queue must be created as a keyed queue. The key for the user queue is described in the format of the user queue entry. The user queue name is passed to the API. The Cluster API documentation contains examples on how to use user queues with Cluster APIs.

When the Distribute Information (QcstDistributeInformation) API is used, the information sent between nodes is deposited on the user queue specified when the CRG was created. This queue must be created by the user on all active nodes in the recovery domain before useing the Distribute Information API. See the Create Cluster Resource Group (QcstCreateClusterResourceGroup) API for details on when the distribute information queue must exist.

The failover message queue receives messages regarding failover activity.

Related concepts

"Maintaining user profiles on all nodes" on page 104
You can use two mechanisms for maintaining user profiles on all nodes within a cluster.

"Determining if a cluster problem exists" on page 140
Start here to diagnose your cluster problems.

Related tasks

"Recovering from cluster job failures" on page 159
Failure of a cluster resource services job is usually indicative of some other problem.

Failover message queue

The failover message queue receives messages regarding failover activity.

Using the failover message queue allows an administrator to be notified before a failover occurs. This gives the administrator the ability to cancel the failover if the desired behavior is to prevent the failover at this time.

The failover message queue is defined when creating a cluster resource group using the Create Cluster (QcstCreateCluster) API. It can also be modified using the CL command and API for changing a cluster resource group. The failover message queue cannot be used with the iSeries Navigator cluster management interface.

Further details on the failover message queue can be found in the Cluster Resource Group API documentation. See the following descriptions for details on how to use the failover message queue.

CL commands

- CRTCRG (Create Cluster Resource Group) command
- CHGCRG (Change Cluster Resource Group) command

APIs

- Create Cluster Resource Group (QcstCreateClusterResourceGroup) API
- Change Cluster Resource Group (QcstChangeClusterResourceGroup) API

Maintaining user profiles on all nodes

You can use two mechanisms for maintaining user profiles on all nodes within a cluster.

| One mechanism is to create a cluster administrative domain to monitor shared resources across nodes in
| a cluster. A cluster administrative domain can monitor several types of resources in addition to user
| profiles, providing easy management of resources that are shared across nodes. See Monitored resources
| for details on these resources. When user profiles are updated, changes are propagated automatically to
| other nodes if the cluster administration domain is active. If the cluster administrative domain is not
| active, the changes will be propagated once the cluster administrative domain is activated.

| With the second mechanism, administrators can also use Management Central in iSeries Navigator to
| perform functions across multiple systems and groups of systems. This support includes some common
| user-administration tasks that operators need to perform across the multiple systems in their cluster.

Management Central allows user profile functions to be performed against groups of systems. The administrator can specify a post-propagation command to be run on the target systems when creating a user profile.

Important:

- If you plan to share user profiles that use password synchronization within a cluster, you must set the Retain Server Security (QRETSVRSEC) system value to 1.
- If you change QRETSVRSEC to 0 after you add an MRE for a user profile, and then change a password (if password is being monitored), the global status for the MRE is set to Inconsistent. The MRE is marked as unusable. Any changes made to the user profile after this change are not synchronized. To recover from this problem, change QRETSVRSEC to 1, remove the MRE, and add the MRE again.

Backup and recovery of clusters

If you use clustering on your systems, it is still important that you create a backup and recovery strategy to protect your data.

If you are planning on using clustering as your backup strategy so that you have one system up and running while the other system is down when its being backed up, it is recommended that you have a minimum of three systems in the cluster. By having three systems in the cluster, you will always have one system to switch over to should a failure occur.

Saving and restoring cluster resource groups

You can save a cluster resource group whether the cluster is active or inactive. The following restrictions apply for restoring a cluster resource group:

- If the cluster is up and the cluster resource group is known to that cluster, you cannot restore the cluster resource group.
- If the node is not configured for a cluster, you cannot restore a cluster resource group.

You can restore a cluster resource group if the cluster is active, the cluster resource group is not known to that cluster, the node is in the recovery domain of that cluster resource group, and the cluster name matches that in the cluster resource group. You can restore a cluster resource group if the cluster is configured but is not active on that node and if that node is in the recovery domain of that cluster resource group.

Preparing for a disaster

In the case of a disaster, you will need to reconfigure your cluster. In order to prepare for such a scenario, it is recommended that you save your cluster configuration information and keep a hardcopy printout of that information.

1. Use the Save Configuration (SAVCFG) command or the Save System (SAVSYS) command after making cluster configuration changes so that the restored internal cluster information is current and consistent with other nodes in the cluster. See Saving configuration information for details on performing a SAVCFG or SAVSYS.
2. Print a copy of cluster configuration information every time you change it. You can use the Display Cluster Information (DSPCLUINF) command to print the cluster configuration. Keep a copy with your backup tapes to use in case of a disaster where you want to reconfigure your entire cluster.

   Related concepts

   "Restoring a cluster from backup tapes" on page 161
   During normal operations, you should never must restore from a backup tape.

   Saving configuration information

"Recovering a cluster after a complete system loss" on page 160
Use this information in conjunction with the appropriate checklist in the Backup and Recovery manual for recovering your entire system after a complete system loss when your server loses power unexpectedly.

"Saving cluster configurations"
You can use commands to save your cluster resource group objects.

"Recovering a cluster after a disaster" on page 161
In the case of a disaster where all your nodes are lost, you will need to reconfigure your cluster.

Related tasks

Planning a backup and recovery strategy

Printing system information

Saving cluster configurations

You can use commands to save your cluster resource group objects.

Use the SAVSYS (Save System) command to save your entire system, not just your configured cluster. You can use the SAVCFG (Save Configuration) command to save your configured system.

SAVOBJ(QUSRSYS/*ALL) OBJTYPE (*CRG)

Note: Cluster resource group objects can be saved only for the current release.

Related tasks

"Backup and recovery of clusters" on page 133
If you use clustering on your systems, it is still important that you create a backup and recovery strategy to protect your data.

Related reference

SAVSYS (Save System) Command

SAVCFG (Save Configuration) Command

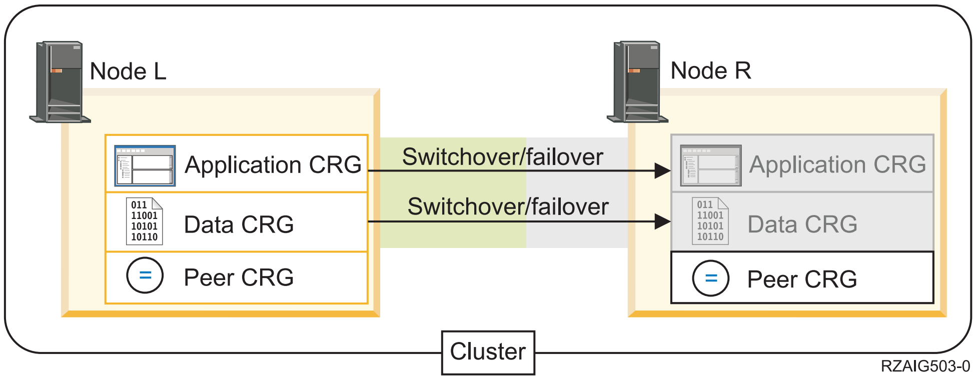
Examples: Cluster configurations

Use these examples of typical cluster implementations to understand when, why, and how using clusters can be beneficial.

Example: A simple, two-node cluster

This configuration example describes a basic cluster that contains two nodes.

This example configuration provides the following:
- One-way replication and failover
- Two-tier environment
- Applications and data move together
- Backup used for offine processing of data
- | Continuous operation on peer CRG



| Using this example, Node L is currently operating as the primary node for two cluster resource groups, an application CRG and a data CRG. It also contains a peer CRG that provide continuous operations for either node. Two exit programs will be running periodically on Node L for the application CRG. The reason two exit programs may be running at the same time is if you call the Start CRG API, an exit program is started and runs continuously while the application CRG is active. If you should call the End CRG API for the application CRG, another exit program is then started. Node R is the first, and only, backup node designated in the recovery domain of each cluster resource group. Data that is associated with the data CRG and pertinent application information that is associated with the application CRG is being replicated from Node L to Node R. If Node L fails or needs to be taken down for administrative reasons, then a failover or switchover is initiated and Node R becomes the primary node for both the application and data CRGs. Node R will take over the Internet Protocol (IP) address defined for the application CRG.

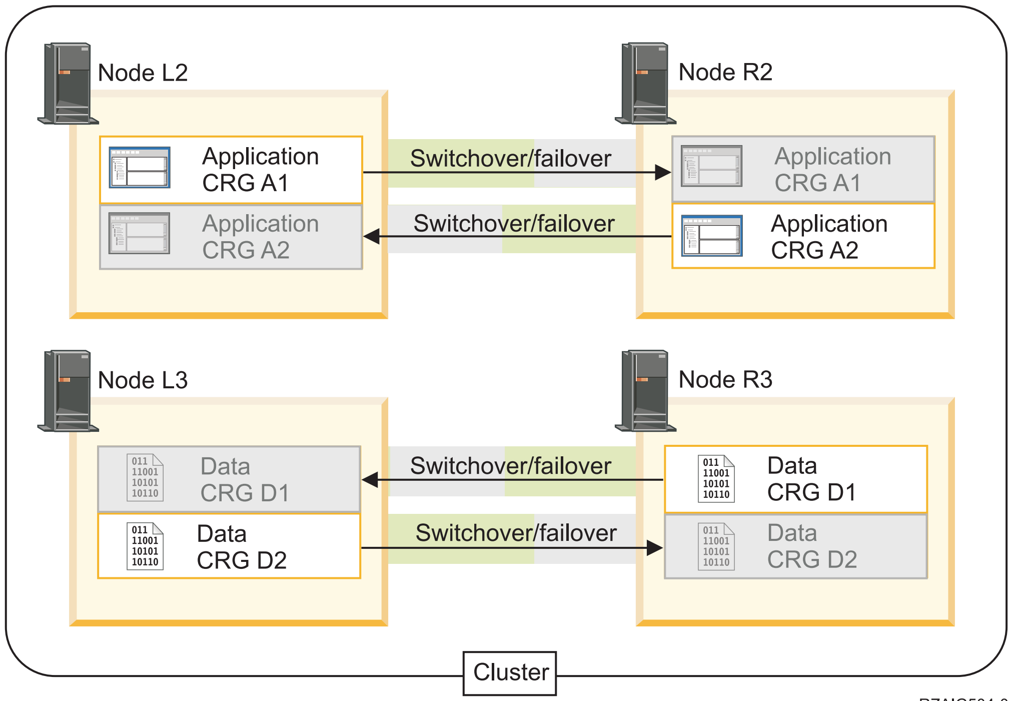
Note: While Node L is down, system availability is exposed because there is no backup should Node R also fail. When Node L recovers and rejoins the cluster, it is made the backup for both cluster resource groups. At that time, replication will be from Node R to Node L. If you want Node L to resume the role of primary, then an administrative switchover should be performed.

Example: A four-node cluster

Use this example to create a more complex cluster that contains four nodes.

This example configuration provides the following:

- Two-way replication and failover
- Three-tier environment
- Applications and data move independently
- Backup is used for normal production of different workload



The four-node example shows the additional flexibility that is possible with an System i cluster. There are two application cluster resource groups (A1 and A2) and two data cluster resource groups (D1 and D2). The data associated with D1 is the critical data for the application associated with A1. The data associated with D2 is the critical data for the application associated with A2. Because this is a three-tier environment, the applications exist on the second tier (Node L2 and Node R2) and the data is separated into the third tier (Node L3 and Node R3).

Cluster resource group (CRG)	Primary	Backup
Application CRG A1	L2	R2
Application CRG A2	R2	L2
Data CRG D1	R3	L3
Data CRG D2	L3	R3

This enables mutual takeover capability at both the application and data levels. All four nodes are being used for normal production. They are also being used to back up other systems in the cluster. The two applications and their associated data should always be available in this cluster. The outage of any single node will not disrupt availability. In addition, the simultaneous outage of a node at the application level with a node at the data level will not disrupt availability.

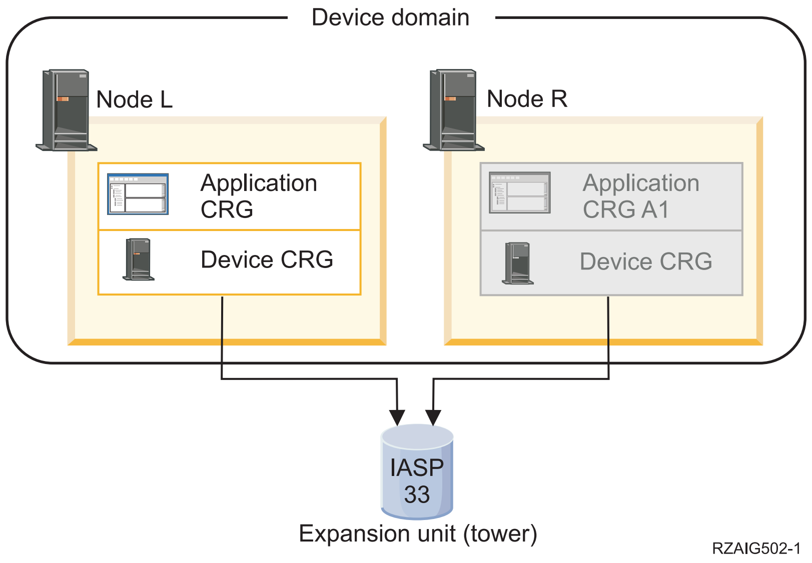
Note: In either instance, the cluster is running exposed in that some cluster resources will not be replicated while a node is down. You can resolve this by having more than one backup for any critical cluster resource.

Example: A switched disk cluster using independent disk pools

A cluster using switched disk technology provides an alternative to having the data replicated. In a switched disk cluster, the data is actually contained in independent disk pools (also referred to as independent ASPs).

This example configuration provides the following:

- One switchable independent disk pool with an idle standby server. The independent disk pool is contained within a collection of disk units that are switchable.
- Two-tier environment
- Applications and data move together
- Backup used for different work loads not associated with this application's data
- No data replication; only one copy of the data exists in this cluster



Using this example, Node L and Node R belong to the same device domain. Node L is currently operating as the primary node for two cluster resource groups - an application CRG and a device CRG. Node R is the first (and only) backup for both of the cluster resource groups. Data that is associated with the device CRG is contained in a switchable resource such as an external expansion unit (tower). Pertinent application information associated with the application CRG is either stored in that same tower or is otherwise being replicated from Node L to Node R. If Node L fails or needs to be taken down for administrative reasons, then Node R becomes the primary node for both cluster resource groups. Node R will take over the Internet Protocol (IP) address defined for the application CRG. Node R will also assume ownership of the switchable resource defined for the device CRG.

Note: While Node L is down, system availability is exposed because there is no backup should Node R also fail. When Node L recovers and rejoins the cluster, it is made the backup for both cluster resource groups. If you want it to again take on the role of primary, then an administrative switchover should be performed.

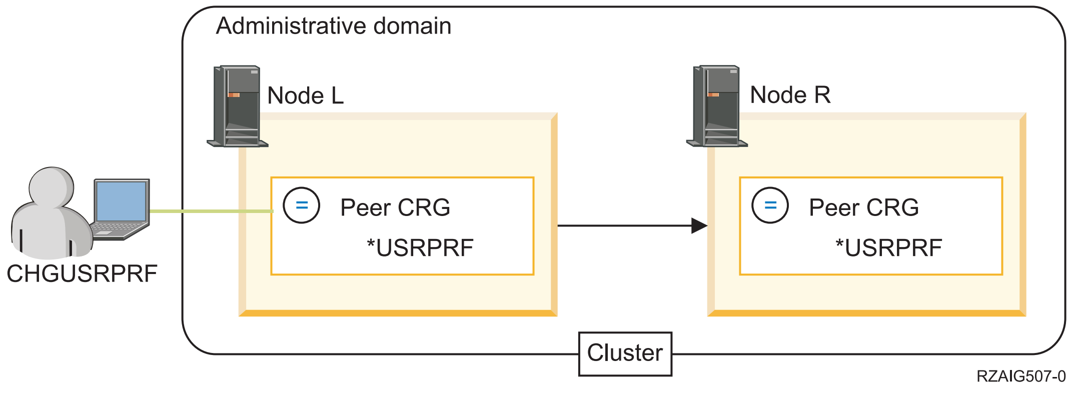
Related concepts

Independent disk pools configurations

Example: A cluster administrative domain to manage peer resources

Provides an example configuration of a cluster administrative domain used to monitor resources within a cluster.

This example configuration provides the following:
- Two-node cluster
- A cluster administrative domain with the two nodes in its domain node list
- A Monitored Resource Entry (MRE) for a user profile which is to be synchronized within the domain.



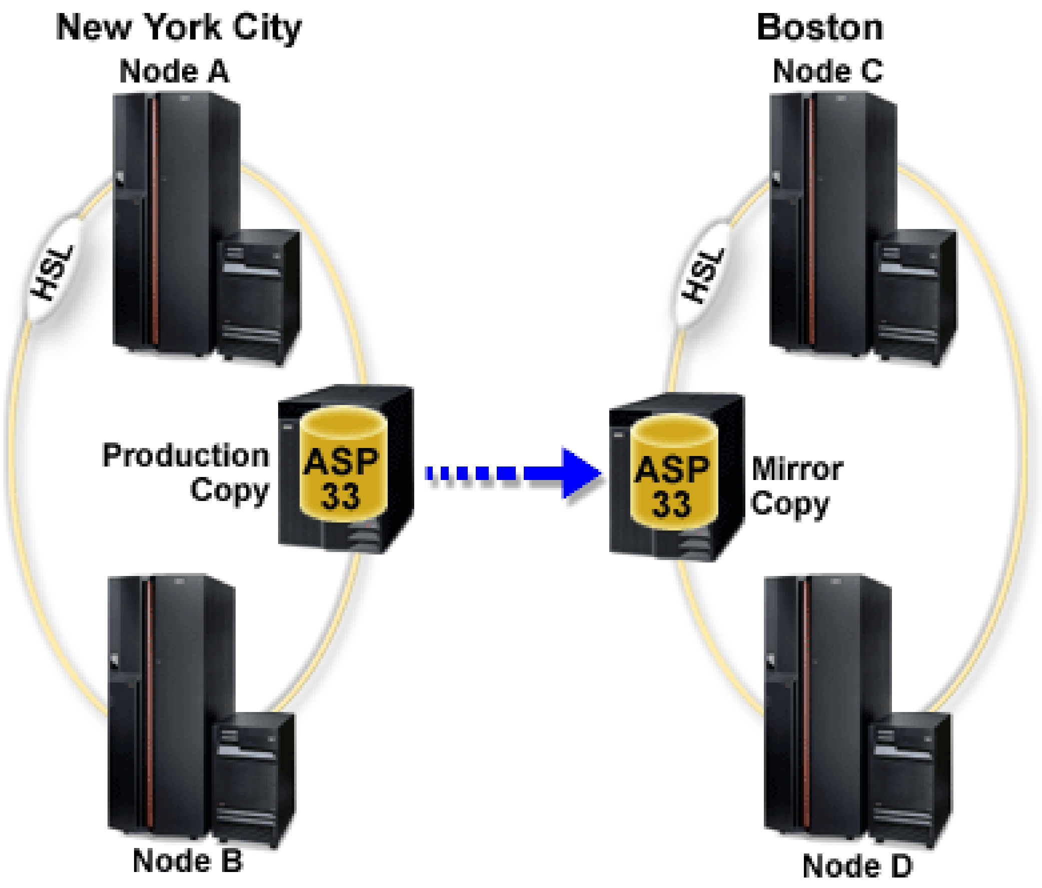
In this example, the administrator wants to ensure the user profile remains consistent throughout the cluster so a cluster administrative domain was created to monitor and synchronize changes to the user profile. The cluster administrative domain is represented by a peer CRG that contains Node L and Node R. A monitored resource entry is added to the cluster administrative domain for the user profile. In this example, all of the attributes of the user profile were specified when the MRE was added. Therefore when any attribute of the user profile is changed on either node L or node R, the change is automatically propagated to the active nodes in the domain, once the CRG is started.

The following steps describe what the administrator completed to set up this example:
1. Create a cluster with Nodes L and R.
2. Create a cluster administrative domain on Nodes L and R
3. Add an MRE to represent the user profile
4. Start the Peer CRG that represents the cluster administrative domain
5. Change the user profile on either node L or on node R. The user profile on the other node will be changed automatically by the cluster administrative domain. The global status for the monitored resource will be consistent if the change is successful.

Example: Independent disk pools with geographic mirroring

The following example shows one way that geographic mirroring can be configured.

Node A and Node B are located in New York City. Node C and Node D are located in Boston. All four nodes are configured in the same recovery domain. The production copy can be switched between nodes A and B. The mirror copy can be switched between nodes C and D. Because all of the nodes are in the same recovery domain, the source system in New York can also exchange roles with the target system in Boston, allowing Boston to host the production copy.



This company has defined the following roles for the nodes in the recovery domain:

Node	Role
Node A	Primary
Node B	Backup 1
Node C	Backup 2
Node D	Backup 3

In the event of a natural disaster in New York, Node C in Boston becomes the primary node by upgrading its mirror copy to a production copy. Node C becomes the source system for geographic mirroring, although geographic mirroring will be suspended because there is no target node because of the natural disaster in New York. When the New York site recovers, Node A becomes a backup node and its previous production copy becomes the mirror copy.

Troubleshooting clusters

Find error recovery solutions for problems that are specific to clusters.

At times, it may appear that the cluster is not working properly. This topic covers information about problems that you may encounter with clusters.

Determining if a cluster problem exists

Start here to diagnose your cluster problems.

At times, it may seem that your cluster is not operating correctly. When you think a problem exists, you can use the following to help determine if a problem exists and the nature of the problem.

- **Determine if clustering is active on your system.**

  To determine if cluster resource services is active, look for the two jobs - QCSTCTL and QCSTCRGM - in the QSYSWRK subsystem. If these jobs are active, then cluster resource services is active. You can use the Work Management function in iSeries Navigator to View jobs in a subsystem or use the WRKACTJOB (Work with Active Jobs) command to do this. You can also use the DSPCLUINF (Display Cluster Information) command to view status information for the cluster.

  – Additional jobs for cluster resource services may also be active. Cluster resource services job structure provides information about how cluster resource services jobs are formatted.

- **Determine the cause of a CPFBB26 message.**

```
Message . . . . : Cluster Resource Services not active or not responding.
Cause . . . . . : Cluster Resource Services is either not active or cannot
respond to this request because a resource is unavailable or damaged.
```

  This error can mean that either the CRG job is not active or the cluster is not active. Use the DSPCLUINF (Display Cluster Information) command to determine if the node is active. If the node is not active, start the cluster node. If it is active, you should also check the CRG to determine whether the CRG has problems.

  Look for the CRG job in the QSYSWRK subsystem. You can use the Work Management function in iSeries Navigator to View jobs in a subsystem or use the WRKACTJOB (Work with Active Jobs) command to do this. You can also use the DSPCRGINF (Display CRG Information) command to view status information for the specific CRG, by specifying the CRG name in the command. If the CRG job is not active, look for the CRG job log to determine the cause of why it was ended. Once the problem is fixed, you could restart the CRG job using CHGCLURCY (Change Cluster Recovery) command or by ending and restarting cluster on that node.

- **Look for messages indicating a problem.**

  – Ensure that you can review all messages associated with a cluster command, by selecting select F10 which toggles between ″Include detailed messages″ and ″Exclude detailed messages″. Select to include all detailed messages and review them to determine if other actions are necessary.

  – Look for inquiry messages in QSYSOPR that are waiting for a response.

  – Look for error messages in QSYSOPR that indicate a cluster problem. Generally, these will be in the CPFBB00 to CPFBBFF range.

  – Display the history log (DSPLOG CL command) for messages that indicate a cluster problem. Generally, these will be in the CPFBB00 to CPFBBFF range.

- **Look at job logs for the cluster jobs for severe errors.**

  These jobs are initially set with a logging level at (4 0 *SECLVL) so that you can see the necessary error messages. You should ensure that these jobs and the exit program jobs have the logging level set appropriately. If clustering is not active, you can still look for spool files for the cluster jobs and exit program jobs.

- **If you suspect some kind of hang condition, look at call stacks of cluster jobs.**

  Determine if there is any program in some kind of DEQW (dequeue wait). If so, check the call stack of each thread and see if any of them have getSpecialMsg in the call stack.

- **Check for cluster vertical licensed internal code (VLIC) logs entries.**

  These log entries have a 4800 major code.

- **Use NETSTAT command to determine if there are any abnormalities in your communications environment.**

NETSTAT returns information about the status of TCP/IP network routes, interfaces, TCP connections and UDP ports on your system.

- Use Netstat Option 1 (Work with TCP/IP interface status) to ensure that the IP addresses chosen to be used for clustering show an 'Active' status. Also ensure that the LOOPBACK address (127.0.0.1) is also active.
- Use Netstat Option 3 (Work with TCP/IP Connection Status) to display the port numbers (F14). Local port 5550 should be in a 'Listen' state. This port must be opened via the STRTCPSVR *INETD command evidenced by the existence of a QTOGINTD (User QTCP) job in the Active Jobs list. If clustering is started on a node, local port 5551 must be opened and be in a '*UDP' state. If clustering is not started, port 5551 must not be opened or it will, in fact, prevent the successful start of clustering on the subject node.

- Use ping. If you try to start a cluster node and it cannot be pinged, you will receive an internal clustering error (CPFBB46).
- **Use the CLUSTERINFO macro to show cluster resource services' view of nodes in the cluster, nodes in the various cluster resource groups, and cluster IP addresses being currently used.**

  Discrepancies found here may help pinpoint trouble areas if the cluster is not performing as expected. See "Investigating a problem with CLUSTERINFO macro" on page 146 for details on using and interpreting the CLUSTERINFO macro results.

Related concepts

"Job structure and user queues" on page 131
When managing cluster, you need to know about job structures and user queues.

Related tasks

View jobs in a subsystem

Related reference

WRKACTJOB (Work with Active Jobs)

DSPCLUINF (Display Cluster Information) command

Gathering recovery information for a cluster

You can use the Work with Cluster (WRKCLU) command to collect information for a complete picture of your cluster. This information can be used to aid in error resolution.

The Work with Cluster (WRKCLU) command is used to display and to work with cluster nodes and objects. When you run this command, the Work with Cluster display is shown. In addition to displaying nodes in a cluster and cluster information, you can use this command to view cluster information and to gather data about your cluster

To gather error recovery information, complete these steps:

1. On a character-based interface, type `WRKCLU OPTION(OPTION)`. You can specify the following options to indicate with which cluster status information you want to work.

   ***SELECT**
   Display the Work with Cluster menu.

   ***NODE**
   Display the Cluster Information panel, which is a list of nodes in the cluster.

   ***CFG** Display the complete configuration parameters for the cluster. It also provides the ability to get detailed information about a cluster resource group.

   ***CRG** Display the list of cluster resource groups in the cluster.

   ***SERVICE**
   Gathers related trace and debug information for all cluster resource service jobs in the cluster.

This information is written to a file with a member for each cluster resource service job. Use this option only when directed by your service provider. It will display a prompt panel for the Dump Cluster Trace (DMPCLUTRC).

Investigating a problem with Dump Cluster Trace (DMPCLUTRC) command

Use the Dump Cluster Trace (DMPCLUTRC) command to help determine and resolve problems with a cluster.

The Dump Cluster Trace (DMPCLUTRC) command can help you to determine whether a cluster job has completed or what the job is currently processing. The command dumps cluster-related trace and debug information to a file. The information is dumped locally on one or more cluster nodes. The command can be used to dump one or all cluster resource service (CRS) jobs. Each CRS job that is dumped has a file member in the file. The name of the file member is the name of the CRS job. Clustering must be active for the command to produce output. Only nodes that have an active CRS job will have output. The information that is dumped originates from the user trace and other information is taken from cluster objects. The amount of information dumped is determined by the dump level. The different dump levels are basic information, error information, informational information, and verbose information. The dump level determines how much information is sent to the file. In most cases an IBM service representative will inform you of which level to specify based on your needs, however; LEVEL(*ERROR) is sufficient for most troubleshooting scenarios. If you have a question on which level is appropriate for your situation, contact an IBM Service representative.

Interpreting trace results

You can analyze the trace results to get an understanding of what the clustering is doing, such as which cluster job is causing the protocol to wait. The output which comes from the user trace will contain a separator line which is a series of equal signs (=). Depending on how many times the DMPCLUTRC was issued will have an impact on how many separator lines are seen in the file. There may be multiple calls of DMPCLUTRC in the same file. The last set of stack dumps has the most current information. In some cases, a CRG job can have two groups. Each group has a separate dump section in the file.

In the following example of Dump Cluster Trace results, there are two nodes (SYSTEM1 and SYSTEM2) in a cluster named MYCLUSTER. It has one CRG named MYCRG. Both nodes are in the recovery domain of the MYCRG. The user issued the STRCRG CL command and the process is taking a long time to return results. From a different workstation, the user entered DMPCLUTRC CLUSTER(MYCLUSTER) CRG(*ALL) LEVEL(*ERROR) FILE(MYFILE) on a command line interface.

For this example, the output from the DMPCLUTRC command resides in a file called MYFILE in member MYCRG. To help explain what the contents of member MYCRG, it has been broken down into sections. Throughout the sections, numbers are highlighted in parentheses to identify the information being described. These details can aid you in troubleshooting cluster problems.

Note: Vertical ellipses indicate that a portion of the trace was removed and is not displayed in the output.

Section 1 of DMPCLUTRC results

```
User Trace Dump for job 073586/QSYS/MYCRG. Size: 300K, Wrapped 0 times.

--- 08/22/2005 16:43:32 ---
(1a) 00000006:658536 Main thread handle 2
(1b) 00000008:748016 Work thread 1 handle 13
(1b) 00000007:754576 Work thread 2 handle 11
--- 08/22/2005 16:46:04 ---
 00000008:269608 CSTDAMBR 1115: WaitForMsg 4 1005 CPFBB3C
--- 08/22/2005 16:48:17 ---
```

```
       00000006:925112
 (1c)       DMPCLUTRC  Node SYSTEM1  Group MYCRG
    ===================================================
```

The first section contains the thread numbers and handles for the cluster job. Cluster jobs may have two or more threads. In this example there is one for the main thread (1a), which is where all the work comes into, and two work threads (1b). This section also contains information about what system this trace came from and to which cluster job it pertains too (1c).

Sections 2 of DMPCLUTRC results

```
00000006:925168 Stack Dump For Target Thread: Handle 2 (0x00000002)
       00000006:925192 Stack:
 (2aa) Main Thread Stack MYCRG
       00000006:925256 Stack:  Library    / Program     Module      Stmt    Procedure
       00000006:933432 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   0     : _CXX_PEP__Fv
       00000006:933488 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   46    : main
       00000006:933536 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   65    : completeStartup__FP8CstDAMbr
       00000006:933584 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   26    : mainQueueProcessLoop__FP8Cs
                                  DAMbr
       00000006:933616 Stack:  QSYS       / QCSTCMN     CSTDAMBR    57    : processQueueMsgs__8CstDAMbrF
                                  Q2_8CstDAMbr13CstQueueIndex
       00000006:933664 Stack:  QSYS       / QCSTCMN     CSTDAMBR    53    : processMsg__8CstDAMbrFP6CstM
                                  sg
       00000006:933712 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
                                  tDAMbr19MsgFunctionPtrEntryP6
       00000006:933744 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   94    : crgDump__FP6CstMsgP8CstDAMbr
       00000006:933792 Stack:  QSYS       / QCSTCMN     CSTACK      95    : CstAckQueryMsg
       00000006:933832 Stack:  QSYS       / QP0ZCPA     QP0ZUDBG    3     : Qp0zDumpTargetStack
00000006:933864 Stack:  QSYS       / QP0ZSCPA    QP0ZSDBG    12    : Qp0zSUDumpTargetStack
       00000006:934016 Stack:  Exception         In Stack Dump Code
       00000006:934040 Stack:  thread is likely terminated or no longer running the same code as the
captured stack
       00000006:934080
 (2a)  Work Thread Index 1  Group MYCRG      Last or current values
 (2e)  00000006:934112   Request handle 8E3E1002 EE3218A1 824F0004 AC000456
 (2c)  00000006:934136   SPI name QcstStartClusterResourceGroup
       00000006:934160 (2g)   POF 10, Completed ack round 1 (2i)
       00000006:934176  (2o) In waitForJobEnd QDFTJOBD  MYCLUSTER     073590
       00000006:934216         Node         Ack Status      POF     (2bb) Nack Msg Id
       00000006:934240 (2n)    SYSTEM1   (2cc) Ready
       00000006:934272     SYSTEM2   Ack    10 (2k)
       00000006:934296   Messages
       00000006:934320 Stack Dump For Target Thread: Handle 13 (0x0000000d)
       00000006:934344 Stack:  Work Thread 1 Stack MYCRG
       00000006:934792 Stack:  Library    / Program     Module      Stmt    Procedure
       00000006:934840 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   9     : workThreadRoutine__FPv
       00000006:934888 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   28    : workQueueProcessLoop__FP8Cst
                                  DAMbr
       00000006:941688 Stack:  QSYS       / QCSTCMN     CSTDAMBR    57    : processQueueMsgs__8CstDAMbrF
                                  Q2_8CstDAMbr13CstQueueIndex
       00000006:941696 Stack:  QSYS       / QCSTCMN     CSTDAMBR    33    : processMsg__8CstDAMbrFP6CstM
                                  sg
       00000006:941712 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
                                  tDAMbr19MsgFunctionPtrEntryP6
       00000006:941728 Stack:  QSYS       / QCSTCMN     CSTACK      3     : CstStripOffHeaderMsgPart
       00000006:941736 Stack:  QSYS       / QCSTCMN     CSTDAMBR    53    : processMsg__8CstDAMbrFP6CstM
                                  sg
       00000006:941752 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
                                  tDAMbr19MsgFunctionPtrEntryP6
       00000006:970888 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    39    : startCrg
       00000006:970912 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    344   : doMessageProcessing__FP6CstM
                                  sgP8CstDAMbr
       00000006:970928 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    57    : doExitPgmPhase__FP6CstMsgP8C
                                  stDAMbr
       00000006:981984 Stack:  QSYS       / QCSTCMN     CSTDAMBR    52    : waitForJobEnd__8CstDAMbrFPA2
```

```
                         6_ci
00000006:982000 Stack:  QSYS       / QCSTCMN     CSTDAMBR    73    : waitForSpecialMsg__8CstDAMbr
                         FP17CstSpecialMsgListPA8_ciT3
00000006:982016 Stack:  QSYS       / QC2UTIL1    QC2MI3      1     : (2dd) deq
00000006:982136 Stack:  Exception         In Stack Dump Code
00000006:982136 Stack:  thread is likely terminated or no longer running the same code as the
captured stack
00000006:982160
(2b)Work Thread Index 2  Group MYCRG      Last or current values
(2f)00000006:982176   Request handle D9C3C8C3 E2E3F5F2 0003 0000
(2cc)00000006:982176   SPI name
00000006:982184 (2h)  POF 0, (2d)Completed ack (2j)round 0
00000006:982184   In getNextWorkMsg
00000006:982208   Node         Ack Status      POF     Nack Msg Id
(2l) 00000006:982208     SYSTEM1   Ready
(2l) 00000006:982232     SYSTEM2   Ready
00000006:982248   Messages
00000006:982256 Stack Dump For Target Thread: Handle 11 (0x0000000b)
00000006:982256 Stack:  Work Thread 2 Stack MYCRG
00000006:982344 Stack:  Library    / Program     Module      Stmt    Procedure
00000006:982360 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   9     : workThreadRoutine__FPv
00000006:982376 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   28    : workQueueProcessLoop__FP8Cst
                         DAMbr
00000006:982392 Stack:  QSYS       / QCSTCMN     CSTDAMBR    51    : processQueueMsgs__8CstDAMbrF
                         Q2_8CstDAMbr13CstQueueIndex
(2m) 00000006:982400 Stack:  QSYS   / QCSTCMN     CSTDAMBR    105   : getNextWorkMsg__8CstDAMbrFv
00000006:982416 Stack:  QSYS       / QC2UTIL1    QC2MI3      1     : deq
00000006:982480 Stack:  Exception        In Stack Dump Code
00000006:982480 Stack:  thread is likely terminated or no longer running the same code as the
captured stack
```

The second section contains the call stacks for each thread that is part of the cluster job. For the most part, the main thread will show the DMPCLUTRC which just completed (2aa). The work threads (2a and 2b) contain the trace information that will help determine what is happening with a cluster job. This section contains details on the call stack, such as the SPI name (2c), the completed acknowledgement (ACK)(2d), request handle for the associated APIs (2e) or the last completed request handle (2f), current point of failure (POF) (2g and 2h), current round of acknowledgement (ACK) (2i and 2j) and the nodes that have acknowledged (ACK) (2k and 2l).

Current *point of failure (POF)* is an internal value that represents where in the code the current protocol is, and may not necessarily indicate a failure has occurred. An *Ack* means the node has successfully completed this part of the protocol and is waiting for all other nodes to acknowledge (ACK) or Nack. A *Nack* means the node cannot successfully complete this part of the protocol and is waiting for all other nodes to respond. The message ID for the Nack is given in the next column (2bb). This is the same message that is sent to the originator's RIQ. If a node fails during a protocol, it's status shows as Fail and, depending on the protocol and the node, may or may not be considered a Nack. An Ack status of Inactive means the node did not participate in the protocol. A value of Ready means that node has not responded back yet. When a thread is in `getNextWorkMsg` (2m) it means the thread is waiting for work to do.

Read the procedure names by starting at the bottom and working up the call stack. This example file contains a `deq` (2dd) with a `waitForSpecialMsg`, `waitForJobEnd` and `doExitPgmPhase`. This indicates that the protocol is waiting for an exit program to complete before it can continue processing. From the `Ack Status` (2k), we can determine which node the protocol is waiting for. In this example, we are waiting for node SYSTEM1 (2n). The qualified job name (2e) indicates the job for which the system is waiting. Once you determine the job name, you can work with the job to troubleshoot the cause of the delay. Some possible causes may be that the job is still waiting in job queue, the job is running but it takes time to process, or the job is waiting for an object that is locked.

In this example, the protocol is waiting for an exit program to complete. An easier way to determine if a protocol is waiting for an exit program or vary job to complete is to look at the first section, and see if in waitForJobEnd (2o). The job name being waited for is on the same line. This eliminates the need to look through the stacks.

Sections 3 of DMPCLUTRC results

```
5722SS1 V5R4M0  060210    AS/400 DUMP    073586/QSYS/MYCRG       08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(3a)OBJ- MYCRGAIX                        CONTEXT- QTEMP
TYPE- *ALL SUBTYPE-*ALL
OBJECT TYPE-          INDEX                                         *CRGM
NAME-        MYCRGAIX                        TYPE-          0E   SUBTYPE-          A5
LIBRARY-     QTEMP     006B8A19B00C9E807000  TYPE-          04   SUBTYPE-          C1
CREATION-    08/22/05  16:43:32              SIZE-          0000007000
ATTRIBUTES-         0000                     ADDRESS-       C7FE286F04 000000
.
.
.
.POINTERS-
  NONE
OIR DATA-
  NONE
END OF DUMP
                                    * * * * *  E N D  O F  L I S T I N G  * * * * *
```

The third section shown is an internal object which contains information about the cluster job. In this example, it's an internal index called MYCRGAIX (3a). The information in here is much easier to read than in section 2 above.

Sections 4 of DMPCLUTRC results

```
5722SS1 V5R4M0  060210   AS/400 DUMP     073586/QSYS/USER     08/22/05 16:48:18       PAGE    1
DMPSYSOBJ PARAMETERS
(4a)OBJ- MYCRGTQ                         CONTEXT- QTEMP
TYPE- 0A   SUBTYPE-EF
OBJECT TYPE-          QUEUE                                         *QTQ
NAME-        MYCRGTQ                         TYPE-          0A   SUBTYPE-          EF
LIBRARY-     QTEMP     006B8A19B00C9E807000  TYPE-          04   SUBTYPE-          C1
CREATION-    08/22/05  16:43:32              SIZE-          000002C000
ATTRIBUTES-         0000                     ADDRESS-       CC6765CAA2 000000
QUEUE ATTRIBUTES-
.
.
.
.POINTERS-
  000010   SPP 1A EF MYCRG     QSYS      073586                                            00002160  0000
  000020   SPP 1A EF MYCRG     QSYS      073586                                            00001540  8000
  000030   SPP 1A EF MYCRG     QSYS      073586                                            000016E0  0000
  000040   SPP 1A EF MYCRG     QSYS      073586                                            00001690  0000
  000050   SPP 1A EF MYCRG     QSYS      073586                                            000016A0  0000
  000070   SPP 1A EF MYCRG     QSYS      073586                                            00002160  0000
OIR DATA-
  NONE
END OF DUMP
                                     * * * * *  E N D  O F  L I S T I N G  * * * * *
```

The fourth section shown is called the trace queue (4a). In this situation it is called MYCRGTQ. This contains information about what the cluster has been having this job execute and how each job responded to the request.

Note: Each message is not described fully here.

Sections 5 of DMPCLUTRC results

```
5722SS1 V5R4M0  060210  AS/400 DUMP  073586/QSYS/MYCRG     08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(5a) OBJ- MYCRG                          CONTEXT- QUSRSYS
TYPE- 19   SUBTYPE-2C
OBJECT TYPE-          SPACE                                         *CRG
NAME-        MYCRG                       TYPE-          19   SUBTYPE-          2C
LIBRARY-     QUSRSYS                     TYPE-          04   SUBTYPE-          01
CREATION-    08/17/05  07:16:40          SIZE-          0000002000
OWNER-       MYCLUSTER                       TYPE-          08   SUBTYPE-          01
ATTRIBUTES-         0800                 ADDRESS-       1EC687A1F3 000000
SPACE ATTRIBUTES-
.
.
.
END OF DUMP
                                          * * * * *  E N D  O F  L I S T I N G  * * * * *
```

The fifth section contains information about the CRG object (5a).

Investigating a problem with CLUSTERINFO macro

The CLUSTERINFO macro displays the information that cluster resource services contains about nodes, CRGs and active cluster IP addresses.

The CLUSTERINFO macro creates a snapshot of information about the current cluster. The command navigates through clustering objects and creates a description of the cluster on the local node. The CLUSTERINFO macro provides a flight recorder for different cluster objects and can aid in determining the source of a problem within the cluster. To access the CLUSTERINFO macro, complete the following steps:

1. On a character-based interface, enter STRSST.
2. Sign on with your Service Tools user profile.
3. On the Start Service Tool, select Option 1 (Start a service tool).
4. Select Option 4 (Display/Alter/Dump).
5. Select Option 2 (Dump to Printer).
6. Select Option 2 (Licensed Internal Code (LIC) data).
7. Select Option 14 (Advanced Analysis).
8. Enter a 1 in front of CLUSTERINFO macro option. Press Enter.

Once the CLUSTERINFO macro displays, use the -H option to display help for all the options that are available with this macro. The following usage diagram describes each of the options that are available for the CLUSTERINFO macro:

Table 30. Options for CLUSTERINFO macro

Option	Description
-H	Displays help screen for options
-A	Displays all information
-FR	Displays flight recorder entries
-HB	Displays heartbeat information
-PERF	Displays performance counters
-Q	Displays outgoing message queue status
-G	Unsuppresses display of all CC broadcast groups
-T	Displays tuning parameters
-M	Displays matrices for all DAs

Table 30. Options for CLUSTERINFO macro (continued)

Option	Description
-DP	Display dataPort information

Interpreting CLUSTERINFO macro results

In this example, the -A option is specified so all fields are dumped. The flight recorders are the main tools for debug. Note that these flight recorders are deleted when the cluster node is ended or deleted. For problem analysis the CLUSTERINFO macro must be run before the cluster is ended or deleted. In some cases the flight recorders may be written to a vlog when the cluster is deleted or ended. The flight recorders record various events that affect the structure and performance of the cluster. It is beyond the scope of this information to provide detailed interpretation of flight recorder data.

Note: Vertical ellipses indicate that a portion of the results was removed and is not displayed in the output.

Section 1 of CLUSTERINFO macro results

```
DISPLAY/ALTER/DUMP    CLUSTERINFO -NEW2     08/23/05  13:36:37   PAGE      1
Running macro: CLUSTERINFO              -A
Use -H for command information
Cluster Name   : MYCLUSTER
Local Node Name: SYSTEM1
CC/CTS Version : 5
Macro Timestamp: 08/23/05 13:36:37.079
```

Section 1 contains generic information about the cluster, such as the cluster name, cluster version, and the timestamp when the report was generated. In this example, the cluster name is MYCLUSTER and the local node name is SYSTEM1.

Section 2 of CLUSTERINFO macro results

```
Cluster Object Addresses
CstcClusterServices Address: DBF08681C9161580
Cluster Address            : FC5B04B0D4001000
Cluster Task Address       : B00010000E932000
Cluster Task Q Address     : DBF08681C9169A00
Clue Group Services Address: CDAB6D0339001000
CC Services Address        : FC5B04B0D4008000
```

Section 2 provides pointers to the locations of key cluster objects.

Section 3 of CLUSTERINFO macro results

```
Message Statistics
Number of non-fragmented messages: 250
Number of fragmented messages    : 1
Number of fragments              : 7
Number of acks                   : 148
```

Section 3 contains messaging statistics for the cluster, such as the number of fragments and the number of acknowledgements (acks).

Section 4 of CLUSTERINFO macro results

```
Node Map
Node ID : SYSTEM1
 GenesisSubnetId : 9.5.251.0
 CCNode *   : FC5B04B0D4007000
 CCSrvNode *: FC5B04B0D404F000
 Adapter 1  : 9.5.251.46 Primary
```

```
  Status    : 0x01 Reachable
  Line Type : 0x09 Ethernet
Node ID : SYSTEM2
 GenesisSubnetId : 9.5.251.0
 CCNode *   : FC5B04B0D4060000
 CCSrvNode *: FC5B04B0D4061000
 Adapter 1  : 9.5.251.47 Primary
  Status    : 0x01 Reachable
  Line Type : 0x09 Ethernet
```

Section 4 lists all the current active cluster nodes in a node map. In this example there are two active nodes, SYSTEM1 and SYSTEM2. T

Section 5 of CLUSTERINFO macro results

```
Subnet Map
Subnet ID: 127.0.0.0
 CCSubnet *           : FC5B04B0D4006000
 CCSrvSubnet*         : FC5B04B0D400C000
 Retries              : 0
 Msg Timeouts         : 0
 Bad Msg Counter      : 0
 Failed Default Address: 0
Subnet ID: 226.5.5.5
 CCSubnet *           : FC5B04B0D405B100
 CCSrvSubnet*         : FC5B04B0D405C000
 Retries              : 0
 Msg Timeouts         : 0
 Bad Msg Counter      : 0
 Failed Default Address: 0
```

Section 5 contains a list of all subnet objects that are in the cluster.

Section 6 of CLUSTERINFO macro results

```
Group Map
Group ID: 0x0000000000000001
 Name  : CTS
 CCGroup *   : FC5B04B0D405FF00
 CCSrvGroup *: FC5B04B0D4064B00
 Member Nodes
  SYSTEM1
  SYSTEM2
Group ID: 0x0000000000000002
 Name  : CTS
 CCGroup *   : FC5B04B0D4055100
 CCSrvGroup *: FC5B04B0D4055200
 Member Nodes
  SYSTEM1
  SYSTEM2
.
.
.
```

Section 6 list of all the current cluster groups. Each group is called a distributed activity group. These groups are used for communication between the groups on each active node in the cluster. Majority of the groups deal are for licensed internal code (LIC). These are identifiable by a group name of CTS and BADA. You will also see a group for CCTL (QCSTCTL job in the operating system), CRGM (QCSTCRGM job in the operating system) and each cluster resource group (CRG) job. Groups for CRG jobs will not have a group name. Each group has member nodes. Member nodes are the nodes which are communicating together for this group.

Section 7 of CLUSTERINFO macro results

```
Partition Map
Partition Map is empty
```

Section 7 contains a list of all the nodes in the SLIC partition list.

Note: This is not the same concept as XPF partitioned nodes.

Section 8 of CLUSTERINFO macro results

```
CTS Client List
CTS Client List is empty
```

Section 8 contains a list of all the registered cluster clients, such as data ports.

Section 9 of CLUSTERINFO macro results

```
Flight Recorder        : CSTCSVFR
Flight Recorder Address: DBF08681C9161620
```

Section 9 contains the cluster services flight recorder (CSTCSVFR) which remains on the system until an IPL.

Section 10 of CLUSTERINFO macro results

```
Message Statistics
Number of non-fragmented messages: 250
Number of fragmented messages    : 1
Number of fragments              : 7
Number of acks                   : 148
Time Stamp: 08/18/05 14:00:15.329
Trace Point: 0x0010 CstcClusterServicesTracePtCreatedFlightRecorder
C3D9C5C1E3C5C6D9          <CREATEFR>
Time Stamp: 08/22/05 16:43:28.912
Trace Point: 0x0020 CstcClusterServicesTracePtCreatedClusterObject
D4D6D9C5E8404040 4040C5F8D3770500  <MYCLUSTER   E8L...>
1000               <..        >
Time Stamp: 08/23/05 13:33:40.935
Trace Point: 0x0030 CstcClusterServicesTracePtDeletedClusterObject
D4D6D9C5E8404040 404040E2E3       <MYCLUSTER      ST  >
Time Stamp: 08/23/05 13:33:41.204
Trace Point: 0x0030 CstcClusterServicesTracePtDeletedClusterObject
C3D4D7E3                          <CMPT            >
Time Stamp: 08/23/05 13:33:55.122
Trace Point: 0x0020 CstcClusterServicesTracePtCreatedClusterObject
D4D6D9C5E8404040 4040FC5B04B0D400 <MYCLUSTER   ....M.>
1000              <..         >

```

Section 10 contains the flight recorder for CSTCCCFR. This cluster flight recorder remains on the system until clustering is ended on this node.

Section 11 of CLUSTERINFO macro results

```
Flight Recorder        : CSTCCLFR
Flight Recorder Address: FC5B04B0D4001E80
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:54.944
Trace Point: 0x1010 CstcClusterTracePtCreatedSubnetObject
7F000000FC5B04B0 D4006000         <........M.-.    >
Time Stamp: 08/23/05 13:33:55.062
Trace Point: 0x1000 CstcClusterTracePtCreatedNodeObject
C3E2E3D9D9C3C8C3 E2E3F5F2FC5B04B0 <CSTRSYSTEM1....>
D4007000                          <M...            >
Time Stamp: 08/23/05 13:33:55.122
```

```
Trace Point: 0x1020 CstcClusterTracePtCreatedMCGroupObject
0000000000000001 00000000D9C3C8C3 <............RCHC>
E2E3F5F2                          <ST52            >
.
.
.
```

Section 11 contains cluster communication flight recorder (CSTECLFR). This cluster flight recorder remains on the system until clustering is ended on this node.

Section 12 of CLUSTERINFO macro results

```
Flight Recorder        : CSTCCCFR
Flight Recorder Address: FC5B04B0D4006380
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:55.080
Trace Point: 0x3000 CstcCCScamTracePtScamOpen
FC5B04B0D400E480 0000000000000000 <....M.U.........>
Time Stamp: 08/23/05 13:33:55.097
Trace Point: 0x3010 CstcCCScamTracePtScamBind
FC5B04B0D400E480 0000000000000000 <....M.U.........>
Time Stamp: 08/23/05 13:33:55.100
Trace Point: 0x3000 CstcCCScamTracePtScamOpen
FC5B04B0D400E480 0000000000000000 <....M.U.........>
D6E4E3                            <OUT             >
Time Stamp: 08/23/05 13:33:55.100
Trace Point: 0x3010 CstcCCScamTracePtScamBind
FC5B04B0D400E480 0000000000000000 <....M.U.........>
.
.
.
```

Section 12 contains the clue flight recorder (CSTCCCFR), which remains on the system until clustering is ended on this node.

Section 13 of CLUSTERINFO macro results

```
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C7E27A7A C3A2A385C7E24082 <CsteGS::CsteGS b>
85878995A2                        <egins           >
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C4C14083 9695A2A399A483A3 <CsteDA construct>
85847A40C2C1C4C1 404040404040     <ed: BADA        >
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C7E27A7A C3A2A385C7E24081 <CsteGS::CsteGS a>
8484408281848140 A39640C4C16D9389 <dd bada to DA_li>
A2A3                              <st              >
.
.
.
```

Section 13 displays the contents of the send queues and active message queues. If this section is not empty for an extended period of time it indicates a problem in the cluster.

Section 14 of CLUSTERINFO macro results

```
Flight Recorder        : CSTECLF2
Flight Recorder Address: CDAB6D0339001300
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C4C17A7A C3A2A385C4C16B40 <CsteDA::CsteDA, >
83998581A385D4C3 C79996A49740C9C4 <createMCGroup ID>
407E40F0A7F1F5                    < = 0x15         >
Time Stamp: 08/23/05 13:33:55.209
C3A2A385E2C3D985 977A7A84859389A5 <CsteSCRep::deliv>
85994094A287E3A8 97857EF0A7F140A2 <er msgType=0x1 s>
```

```
A482E3A897857EF0 A7F240C4C17EC2C1 <ubType=0x2 DA=BA>
C4C1404040404040                  <DA              >
Time Stamp: 08/23/05 13:33:55.209
C3A2A385C4C17A7A A58985A66B409985 <CsteDA::view, re>
9496A585D4C3C799 96A497D485948285 <moveMCGroupMembe>
99A240C9C4407E40 F0A7F1F540969384 <rs ID = 0x15 old>
6D959684856D9389 A2A340A289A98540 <_node_list size >
7E40F0A7F0                        <= 0x0           >
.
.
.
```

Section 14 contains flight recorder information.

Section 15 of CLUSTERINFO macro results

```
Message Queues
Send Queues:
Send Queue: 00 Messages: 00 MessageQueue*: FC5B04B0D400BF80
Send Queue: 01 Messages: 00 MessageQueue*: FC5B04B0D400DF80
Send Queue: 02 Messages: 00 MessageQueue*: FC5B04B0D400E600
Send Queue: 03 Messages: 00 MessageQueue*: FC5B04B0D400E680
Send Queue: 04 Messages: 00 MessageQueue*: FC5B04B0D400E700
Send Queue: 05 Messages: 00 MessageQueue*: FC5B04B0D400E780
Send Queue: 06 Messages: 00 MessageQueue*: FC5B04B0D400E800
Send Queue: 07 Messages: 00 MessageQueue*: FC5B04B0D400E880
Send Queue: 08 Messages: 00 MessageQueue*: FC5B04B0D400E900
Send Queue: 09 Messages: 00 MessageQueue*: FC5B04B0D400E980
Send Queue: 10 Messages: 00 MessageQueue*: FC5B04B0D400EA00
Send Queue: 11 Messages: 00 MessageQueue*: FC5B04B0D400EA80
Send Queue: 12 Messages: 00 MessageQueue*: FC5B04B0D400EB00
Send Queue: 13 Messages: 00 MessageQueue*: FC5B04B0D400EB80
Send Queue: 14 Messages: 00 MessageQueue*: FC5B04B0D400EC00
Send Queue: 15 Messages: 00 MessageQueue*: FC5B04B0D400EC80
Send Queue: 16 Messages: 00 MessageQueue*: FC5B04B0D400ED00
Send Queue: 17 Messages: 00 MessageQueue*: FC5B04B0D400ED80
Send Queue: 18 Messages: 00 MessageQueue*: FC5B04B0D400EE00
Send Queue: 19 Messages: 00 MessageQueue*: FC5B04B0D400EE80
Active Message Queues:
Active Message Queue: 00 Messages: 00 MessageQueue*: FC5B04B0D4008570
Active Message Queue: 01 Messages: 00 MessageQueue*: FC5B04B0D4008640
Active Message Queue: 02 Messages: 00 MessageQueue*: FC5B04B0D4008710
Active Message Queue: 03 Messages: 00 MessageQueue*: FC5B04B0D40087E0
Active Message Queue: 04 Messages: 00 MessageQueue*: FC5B04B0D40088B0
Active Message Queue: 05 Messages: 00 MessageQueue*: FC5B04B0D4008980
Active Message Queue: 06 Messages: 00 MessageQueue*: FC5B04B0D4008A50
Active Message Queue: 07 Messages: 00 MessageQueue*: FC5B04B0D4008B20
Active Message Queue: 08 Messages: 00 MessageQueue*: FC5B04B0D4008BF0
Active Message Queue: 09 Messages: 00 MessageQueue*: FC5B04B0D4008CC0
Active Message Queue: 10 Messages: 00 MessageQueue*: FC5B04B0D4008D90
Active Message Queue: 11 Messages: 00 MessageQueue*: FC5B04B0D4008E60
Active Message Queue: 12 Messages: 00 MessageQueue*: FC5B04B0D4008F30
Active Message Queue: 13 Messages: 00 MessageQueue*: FC5B04B0D4009000
Active Message Queue: 14 Messages: 00 MessageQueue*: FC5B04B0D40090D0
Active Message Queue: 15 Messages: 00 MessageQueue*: FC5B04B0D40091A0
Active Message Queue: 16 Messages: 00 MessageQueue*: FC5B04B0D4009270
Active Message Queue: 17 Messages: 00 MessageQueue*: FC5B04B0D4009340
Active Message Queue: 18 Messages: 00 MessageQueue*: FC5B04B0D4009410
Active Message Queue: 19 Messages: 00 MessageQueue*: FC5B04B0D40094E0
--------------------------------------------------------------------------------
Tuning Values
cstcRcvSendTimerRatio        : 2 Default: 2
cstcMcastRelayTimerRatio     : 8 Default: 8
cstcMcastRelayHBTimerRatio   : 4 Default: 4
cstcHeartbeatBaseTimer       : 12288000000 Default: 12288000000
cstcHeartbeatBasePrecision   : 4096000000 Default: 4096000000
cstcRetryPrecision           : 4096000000 Default: 4096000000
```

```
cstcRetryTimerVal            : 4096000000 Default: 4096000000
cstcCDATBaseTimer            : 491520000000 Default: 491520000000
cstcCDATBasePrecision        : 40960000000 Default: 40960000000
cstcRecoveryBaseTimer        : 3686400000000 Default: 3686400000000
cstcRecoveryBasePrecision    : 491520000000 Default: 491520000000
cstcMaxRetryTime             : 32768000000 Default: 32768000000
cstcCCFragmentSize           : 1464 Default: 1464
cstcCCSendQOverflow          : 1024 Default: 1024
cstcBadMsgCtrThreshold       : 3 Default: 3
cstcUnreachableHBAckThreshold: 1 Default: 1
cstcReachableHBAckThreshold  : 3 Default: 3
cstcUnreachableHBThreshold   : 4 Default: 4
cstcReachableHBThreshold     : 4 Default: 4
cstcMaxHBThreshold           : 16 Default: 16
cstcDisableMsgTimer          : 0 Default: 0
cstcRepeatAckThreshold       : 10 Default: 10
     DISPLAY/ALTER/DUMP    CLUSTERINFO -NEW2   08/23/05  13:36:37   PAGE     87
cstcDelayedAckTimer          : 409600000 Default: 409600000
cstcDelayedAckPrecision      : 40960000 Default: 40960000
cstcCCSendWindow             : 2 Default: 2
cstcCCEnableMcast            : 1 Default: 1
cstcCCPerfClass              : 2 Default: 2
-------------------------------------------------------------------------------
****** END OF DUMP ******
```

Section 15 contains tuning values. Tuning values are equivilent to the cluster performance and configuration information that is described in the Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo) API. Section 14 contains the current value and the default for these fields.

Common cluster problems

Lists some of the most common problems that can occur in a cluster, as well as ways to avoid and recover from them.

The following common problems are easily avoidable or easily correctable.

You cannot start or restart a cluster node

This situation is typically due to some problem with your communications environment. To avoid this situation, ensure that your network attributes are set correctly, including the loopback address, INETD settings, ALWADDCLU attibute, and the IP addresses for cluster communications.

- The ALWADDCLU network attribute must be appropriately set on the target node if trying to start a remote node. This should be set to either *ANY or *RQSAUT depending on your environment.
- The IP addresses chosen to be used for clustering locally and on the target node must show an *Active* status.
- The LOOPBACK address (127.0.0.1) locally and on the target node must also be active.
- The local and any remote nodes must be able to PING using the IP addresses to be used for clustering to insure network routing is active.
- INETD must be active on the target node. When INETD is active, port 5550 on the target node should be in a *Listen* state. See INETD server for information about starting the INETD server.
- Prior to attempting to start a node, port 5551 on the node to be started must not be opened or it will, in fact, prevent the successful start of clustering on the subject node.

You end up with several, disjointed one-node clusters

This can occur when the node being started cannot communicate with the rest of the cluster nodes. Check the communications paths.

The response from exit programs is slow.

A common cause for this situation is incorrect setting for the job description used by the exit program. The MAXACT parameter may be set too low so that, for example, only one instance of the exit program can be active at any point in time. It is recommended that this be set to *NOMAX.

Performance in general seems to be slow.

There are several common causes for this symptom.

- The most likely cause is heavy communications traffic over a shared communications line.
- Another likely cause is an inconsistency between the communications environment and the cluster message tuning parameters. You can use the Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo) API to view the current settings of the tuning parameters and the Change Cluster Resource Services (QcstChgClusterResourceServices) API to change the settings. Cluster performance may be degraded under default cluster tuning parameter settings if using old adapter hardware. The adapter hardware types included in the definition of *old* are 2617, 2618, 2619, 2626, and 2665. In this case, setting of the *Performance class* tuning parameter to *Normal* is desired.
- Another common cause of this condition is problems with the IP multicast groups. If the primary cluster addresses (first address entered for a given node when creating a cluster or adding a node) for several nodes reside on a common LAN, the cluster will utilize IP multicast capability. Using the NETSTAT command, insure the primary cluster addresses show a multicast host group of 226.5.5.5. This can be seen using option 14 *Display multicast group* for the subject address. If the multicast group does not exist, verify the default setting of TRUE is still set for the *Enable multicast* cluster tuning parameter by using the Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo) API.
- If all the nodes of a cluster are on a local LAN or have routing capabilities which can handle Maximum Transmission Unit (MTU) packet sizes of greater than 1,464 bytes throughout the network routes, large cluster message transfers (greater than 1,536K bytes) can be greatly speeded up by increasing the cluster tuning parameter value for *Message fragment size* to better match the route MTUs.

You cannot use any of the function of the new release.

If you attempt to use new release function and you see error message CPFBB70, then your current cluster version is still set at the prior version level. You must upgrade all cluster nodes to the new release level and then use the adjust cluster version interface to set the current cluster version to the new level. See Adjust the cluster version of a cluster for more information.

You cannot add a node to a device domain or access the iSeries Navigator cluster management interface.

To access the iSeries Navigator cluster management interface, or to use switchable devices, you must have i5/OS Option 41, HA Switchable Resources installed on your system. You must also have a valid license key for this option.

You applied a cluster PTF and it does not seem to be working.

| You should ensure that you have completed the following tasks after applying the PTF:

1. End the cluster
2. Signoff then signon

   The old program is still active in the activation group until the activation group is destroyed. All of the cluster code (even the cluster APIs) run in the default activation group.
3. Start the cluster

   Most cluster PTFs require clustering to be ended and restarted on the node to activate the PTF.

CEE0200 appears in the exit program joblog.

On this error message, the from module is QLEPM and the from procedure is Q_LE_leBdyPeilog. Any program that the exit program invokes must run in either *CALLER or a named activation group. You must change your exit program or the program in error to correct this condition.

CPD000D followed by CPF0001 appears in the cluster resource services joblog.

When you receive this error message, make sure the QMLTTHDACN system value is set to either 1 or 2.

Cluster appears hung.

Make sure cluster resource group exit programs are outstanding. To check the exit program, use the WRKACTJOB (Work with Active Jobs) command, then look in the Function column for the presence of PGM-QCSTCRGEXT.

Related concepts

"Enabling a node to be added to a cluster" on page 103
Before you can add a node to a cluster, you need to set a value for the Allow add to cluster (ALWADDCLU) network attribute.

"Cluster performance" on page 127
When changes are made to a cluster, the overhead necessary to manage the cluster can be affected.

"Cluster version" on page 20
A *cluster version* represents the level of function available on the cluster.

"iSeries Navigator cluster management" on page 83
IBM offers a cluster management interface that is available through iSeries Navigator and accessible through Option 41 (i5/OS - HA Switchable Resources).

Related tasks

"Adjusting the cluster version of a cluster" on page 115
The cluster version defines the level at which all the nodes in the cluster are actively communicating with each other.

Partition errors

Certain cluster conditions are easily corrected. If a cluster partition has occurred, you can learn how to recover. This topic also tells you how to avoid a cluster partition and gives you an example of how to merge partitions back together.

A cluster partition occurs in a cluster whenever contact is lost between one or more nodes in the cluster and a failure of the lost nodes cannot be confirmed. This is not to be confused with a partition in a logical partition (LPAR) environment.

If you receive error message CPFBB20 in either the history log (QHST) or the QCSTCTL joblog, a cluster partition has occurred and you need to know how to recover. The following example shows a cluster partition that involves a cluster made up of four nodes: A, B, C, and D. The example shows a loss of communication between cluster nodes B and C has occurred, which results in the cluster dividing into two cluster partitions. Before the cluster partition occurred, there were four cluster resource groups, which can be of any type, called CRG A, CRG B, CRG C, and CRG D. The example shows the recovery domain of each cluster resource group.

Table 31. Example of a recovery domain during a cluster partition

Node A	Node B	x	Node C	Node D
CRG A (backup1)	CRG A (primary)			
	CRG B (primary)		CRG B (backup1)	
	CRG C (primary)		CRG C (backup1)	CRG C (backup2)
CRG D (backup2)	CRG D (primary)		CRG D (backup1)	
Partition 1			**Partition 2**	

A cluster may partition if the maximum transmission unit (MTU) at any point in the communication path is less than the cluster communications tuneable parameter, message fragment size. MTU for a cluster IP address can be verified using the Work with TCP/IP Network Status (WRKTCPSTS) command on the subject node. The MTU must also be verified at each step along the entire communication path. If the MTU is less than the message fragment size, either raise the MTU of the path or lower the message fragment size. You can use the Retrieve Cluster Resource Services Information (QcstRetrieveCRSInfo) API to view the current settings of the tuning parameters and the Change Cluster Resource Services (QcstChgClusterResourceServices) API to change the settings.

Once the cause of the cluster partition condition has been corrected, the cluster will detect the re-established communication link and issue the message CPFBB21 in either the history log (QHST) or the QCSTCTL joblog. This informs the operator that the cluster has recovered from the cluster partition. Be aware that once the cluster partition condition has been corrected, it may be a few minutes before the cluster merges back together.

Related concepts

"Cluster partition" on page 37
A *cluster partition* is a subset of the active cluster nodes that results from a communications failure. Members of a partition maintain connectivity with each other.

"Avoiding a cluster partition" on page 96
The typical network-related cluster partition can best be avoided by configuring redundant communications paths between all nodes in the cluster.

"Merge" on page 32
A *merge* operation is similar to a rejoin operation except that it occurs when nodes that are partitioned begin communicating again.

"Example: Failure" on page 27
Usually, a failover results from a node failure, but there are other reasons that can also generate a failover.

Determining primary and secondary cluster partitions

| In order to determine the types of cluster resource group actions that you can take within a cluster
| partition, you need to know whether the partition is a primary or a secondary cluster partition. When a
| partition is detected, each partition is designated as a primary or secondary partition for each cluster
| resource group defined in the cluster.

| For primary-backup model, the primary partition contains the node that has the current node role of
| primary. All other partitions are secondary. The primary partition may not be the same for all cluster
| resource groups.

| A peer model has the following partition rules:

| • If the recovery domain nodes are fully contained within one partition, it will be the primary partition.
| • If the recovery domain nodes span a partition, there will be no primary partition. Both partitions will
| be secondary partitions.

- If the cluster resource group is active and there are no peer nodes in the given partition, the cluster resource group will be ended in that partition.
- Operational changes are allowed in a secondary partition as long as the restrictions for the operational changes are met.
- No configuration changes are allowed in a secondary partition.

The restrictions for each Cluster Resource Group API are:

Table 32. Cluster Resource Group API Partition Restrictions

Cluster Resource Group API	Allowed in primary partition	Allowed in secondary partitions
Add Node to Recovery Domain	X	
Add CRG Device Entry		
Change Cluster Resource Group	X	
Change CRG Device Entry	X	X
Create Cluster Resource Group		
Delete Cluster Resource Group	X	X
Distribute Information	X	X
End Cluster Resource Group[1]	X	
Initiate Switchover	X	
List Cluster Resource Groups	X	X
List Cluster Resource Group Information	X	X
Remove Node from Recovery Domain	X	
Remove CRG Device Entry	X	
Start Cluster Resource Group[1]	X	

Note:

1. Allowed in all partitions for peer cluster resource groups, but only affects the partition running the API.

By applying these restrictions, cluster resource groups can be synchronized when the cluster is no longer partitioned. As nodes rejoin the cluster from a partitioned status, the version of the cluster resource group in the primary partition is copied to nodes from a secondary partition.

When merging two secondary partitions for peer model, the partition which has cluster resource group with status of Active will be declared the winner. If both partitions have the same status for cluster resource group, the partition which contains the first node listed in the cluster resource group recovery domain will be declared the winner. The version of the cluster resource group in the winning partition will be copied to nodes in another partition.

When a partition is detected, the Add Cluster Node Entry, Adjust Cluster Version, and the Create Cluster API cannot be run in any of the partitions. The Add Device Domain Entry API can only be run if none of the nodes in the device domain are partitioned. All of the other Cluster Control APIs may be run in any partition. However, the action performed by the API takes affect only in the partition running the API.

Changing partitioned nodes to failed

Sometimes, a partitioned condition is reported when there really was a node outage. This can occur when cluster resource services loses communications with one or more nodes, but cannot detect if the nodes are still operational. When this condition occurs, a simple mechanism exists for you to indicate that the node has failed.

Attention: When you tell cluster resource services that a node has failed, it makes recovery from the partition state simpler. However, changing the node status to failed when, in fact, the node is still active and a true partition has occurred should not be done. Doing so can cause a node in more than one partition to assume the primary role for a cluster resource group. When two nodes think they are the primary node, data such as files or databases can become disjoint or corrupted if multiple nodes are each independently making changes to their copies of files. In addition, the two partitions cannot be merged back together when a node in each partition has been assigned the primary role.

When the status of a node is changed to Failed, the role of nodes in the recovery domain for each cluster resource group in the partition may be reordered. The node being set to Failed will be assigned as the last backup. If multiple nodes have failed and their status needs to be changed, the order in which the nodes are changed will affect the final order of the recovery domain's backup nodes. If the failed node was the primary node for a CRG, the first active backup will be reassigned as the new primary node.

Related concepts

"Merge" on page 32
A *merge* operation is similar to a rejoin operation except that it occurs when nodes that are partitioned begin communicating again.

"Rejoin" on page 30
Rejoin means to become an active member of a cluster after having been a nonparticipating member.

Related tasks

"Tips: Cluster partitions" on page 158
Use these tips for cluster partitions.

Related reference

CHGCLUNODE command

Change Cluster Node Entry API (QcstChangeClusterNodeEntry)

STRCLUNOD command

Start Cluster Node (QcstStartClusterNode) API

Using iSeries Navigator:

This requires Option 41 (i5/OS - HA Switchable Resources) to be installed and licensed.

When cluster resource services has lost communications with a node but cannot detect if the node is still operational, a cluster node will have a status of **Not communicating** in the Nodes container in iSeries Navigator. You may need to change the status of the node from **Not communicating** to **Failed**. You will then be able to restart the node.

To change the status of a node from **Not communicating** to **Failed**, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Expand the cluster that contains the node for which you want to change the status.
4. Click **Nodes**.
5. Right-click the node for which you want to change the status, and select **Cluster** → **Change Status**.

and select ClusterChange Status

To restart the node, follow these steps:
1. Right-click the node, and select **Cluster** → **Start**.

Using CL commands and APIs:
To change the status of a node from **Not communicating** to **Failed**, follow these steps:

1. Use the CHGCLUNODE command or the Change Cluster Node Entry (QcstChangeClusterNodeEntry) API to change the status of a node from partitioned to failed. This should be done for all nodes that have actually failed.
2. Use the STRCLUNOD command or the Start Cluster Node (QcstStartClusterNode) API to start the cluster node, allowing the node to rejoin the cluster.

Tips: Cluster partitions

Use these tips for cluster partitions.

1. The rules for restricting operations within a partition are designed to make merging the partitions feasible. Without these restrictions, reconstructing the cluster requires extensive work.
2. If the nodes in the primary partition have been destroyed, special processing may be necessary in a secondary partition. The most common scenario that causes this condition is the loss of the site that made up the primary partition. Use the example in recovering from partition errors and assume that Partition 1 was destroyed. In this case, the primary node for Cluster Resource Groups B, C, and D must be located in Partition 2. The simplest recovery is to use Change Cluster Node Entry to set both Node A and Node B to failed. See changing partitioned nodes to failed for more information about how to do this. Recovery can also be achieved manually. In order to do this, perform these operations:
   a. Remove Nodes A and B from the cluster in Partition 2. Partition 2 is now the cluster.
   b. Establish any logical replication environments needed in the new cluster. IE. Start Cluster Resource Group API/CL command, and so on.

   Since nodes have been removed from the cluster definition in Partition 2, an attempt to merge Partition 1 and Partition 2 will fail. In order to correct the mismatch in cluster definitions, run the Delete Cluster (QcstDeleteCluster) API on each node in Partition 1. Then add the nodes from Partition 1 to the cluster, and reestablish all the cluster resource group definitions, recovery domains, and logical replication. This requires a great deal of work and is also prone to errors. It is very important that you do this procedure only in a site loss situation.
3. Processing a start node operation is dependent on the status of the node that is being started:

   The node either failed or an End Node operation ended the node:
   a. Cluster resource services is started on the node that is being added
   b. Cluster definition is copied from an active node in the cluster to the node that is being started.
   c. Any cluster resource group that has the node being started in the recovery domain is copied from an active node in the cluster to the node being started. No cluster resource groups are copied from the node that is being started to an active node in the cluster.

   The node is a partitioned node:
   a. The cluster definition of an active node is compared to the cluster definition of the node that is being started. If the definitions are the same, the start will continue as a merge operation. If the definitions do not match, the merge will stop, and the user will need to intervene.
   b. If the merge continues, the node that is being started is set to an active status.
   c. Any cluster resource group that has the node being started in the recovery domain is copied from the primary partition of the cluster resource group to the secondary partition of the cluster resource group. Cluster resource groups may be copied from the node that is being started to nodes that are already active in the cluster.

Related tasks

“Changing partitioned nodes to failed” on page 156
Sometimes, a partitioned condition is reported when there really was a node outage. This can occur when cluster resource services loses communications with one or more nodes, but cannot detect if the nodes are still operational. When this condition occurs, a simple mechanism exists for you to indicate that the node has failed.

Related reference

Delete Cluster (QcstDeleteCluster) API

Cluster recovery

Read about how to recover from other cluster failures that may occur.

Recovering from cluster job failures

Failure of a cluster resource services job is usually indicative of some other problem.

You should look for the job log associated with the failed job and look for messages that describe why it failed. Correct any error situations.

| You can use the Change Cluster Recovery (CHGCLURCY) command to restart a cluster resource group
| job that was ended without having to end and restart clustering on a node.

| 1. `CHGCLURCY CLUSTER(EXAMPLE)CRG(CRG1)NODE(NODE1)ACTION(*STRCRGJOB)` This command will cause
| cluster resource group job, CRG1, on node NODE1 to be submitted. To start the cluster resource
| group job on NODE1 requires clustering to be active on NODE1.
2. Restart clustering on the node.

If you are using a IBM Business Partner cluster management product, refer to the documentation that came with the product.

Related concepts

"Job structure and user queues" on page 131
When managing cluster, you need to know about job structures and user queues.

Related tasks

"Ending a cluster node" on page 114
Stopping or ending a node stops cluster resource services on that node.

"Starting a cluster node" on page 113
Starting a cluster node starts cluster resource services on a node in the cluster. Beginning with cluster version 3, a node can start itself and will be able to rejoin the current active cluster, provided it can find an active node in the cluster.

Recovering a damaged cluster object

While it is unlikely you will ever experience a damaged object, it may be possible for cluster resource services objects to become damaged.

The system, if it is an active node, will attempt to recover from another active node in the cluster. The system will perform the following recovery steps:

For a damaged internal object

1. The node that has the damage ends.
2. If there is at least one other active node within the cluster, the damaged node will automatically restart itself and rejoin the cluster. The process of rejoining will correct the damaged situation.

For a damaged cluster resource group

1. The node that has a damaged CRG will fail any operation currently in process that is associated with that CRG. The system will then attempt to automatically recover the CRG from another active node.
2. If there is at least one active member in the recovery domain, the CRG recovery will work. Otherwise, the CRG job ends.

If the system cannot identify or reach any other active node, you will need to perform these recovery steps.

For a damaged internal object

You receive an internal clustering error (CPFBB46, CPFBB47, or CPFBB48).

1. End clustering for the node that contains the damage.
2. Restart clustering for the node that contains the damage. Do this from another active node in the cluster.
3. If Steps 1 and 2 do not solve the problem, remove the damaged node from the cluster.
4. Add the system back into the cluster and into the recovery domain for the appropriate cluster resource groups.

For a damaged cluster resource group

You receive an error stating that an object is damaged (CPF9804).

1. End clustering on the node that contains the damaged cluster resource group.
2. Delete the CRG (using the DLTCRG command).
3. If there is no other node active in the cluster that contains the CRG object, restore from media.
4. Start clustering on the node that contains the damaged cluster resource group. This can be done from any active node.
5. When you start clustering, the system resynchronizes all of the cluster resource groups. You may need to recreate the CRG if no other node in the cluster contains the CRG.

Recovering a cluster after a complete system loss

Use this information in conjunction with the appropriate checklist in the Backup and Recovery manual for recovering your entire system after a complete system loss when your server loses power unexpectedly.

Scenario 1: Restoring to the same system

1. In order to prevent inconsistencies in the device domain information between the Licensed Internal Code and i5/OS, it is recommended that you install the Licensed Internal Code using option 3 (Install Licensed Internal Code and Recover Configuration).

   Note: For the Install Licensed Internal Code and Recover Configuration operation to succeed, you must have the same disk units -- with exception of the load source disk unit if it has failed. You must also be recovering the same release.
2. After you have installed the Licensed Internal Code, follow the *How to Recover Your Disk Configuration* procedure in chapter 5 of the *Backup and Recovery* manual. These steps will help you avoid having to reconfigure the ASPs.
3. After you have recovered your system information and are ready to start clustering on the node you just recovered, you must start clustering from the active node. This will propagate the most current configuration information to the recovered node.

Scenario 2: Restoring to a different system

After you have recovered your system information and checked the job log to make sure that all objects have restored, you must perform the following steps to obtain the correct cluster device domain configuration.

1. From the node you just restored, delete the cluster.
2. From the active node, perform these steps:
   a. Remove the recovered node from the cluster.
   b. Add the recovered node back into the cluster.
   c. Add the recovered node to the device domain.
   d. Create the cluster resource group or add the node to the recovery domain.

Related tasks

"Backup and recovery of clusters" on page 133
If you use clustering on your systems, it is still important that you create a backup and recovery strategy to protect your data.

Related information

Backup and Recovery

Recovering a cluster after a disaster

In the case of a disaster where all your nodes are lost, you will need to reconfigure your cluster.

In order to prepare for such a scenario, it is recommended that you save your cluster configuration information and keep a hardcopy printout of that information.

Related tasks

"Backup and recovery of clusters" on page 133
If you use clustering on your systems, it is still important that you create a backup and recovery strategy to protect your data.

Restoring a cluster from backup tapes

During normal operations, you should never must restore from a backup tape.

This is only necessary when a disaster occurs and all nodes were lost in your cluster. If a disaster should occur, you recover by following your normal recovery procedures that you have put in place after you created your backup and recovery strategy.

Related tasks

"Backup and recovery of clusters" on page 133
If you use clustering on your systems, it is still important that you create a backup and recovery strategy to protect your data.

Related information

Backup and Recovery

Frequently asked questions about iSeries Navigator cluster management

Questions and answers about the iSeries Navigator graphical user interface for creating and managing clusters.

The IBM graphical user interface for creating and managing clusters is available in iSeries Navigator and accessible through Option 41 (HA Switchable Resources). See iSeries Navigator cluster management for details on the interface.

Here is a list of iSeries Navigator cluster management questions and answers.

General

1. Is there a checklist that outlines the prerequisites to creating a cluster?

iSeries Navigator cluster management

1. Where is the Clusters function located in the iSeries Navigator interface?
2. How do I create a cluster?
3. What is the relationship between the Clusters folder and the Management Central system group?
4. I already have a cluster defined on some iSeries systems in the network. How do I add it so I can view and manage it through iSeries Navigator?
5. None of the nodes in my cluster have a status of ″Started″. Which node should I start first?

6. Why should I care which node is started first?
7. What does the ″Current Primary Node″ column mean in the switchable devices and switchable applications folders?
8. How do I find a device cluster resource group (CRG) in iSeries Navigator?
9. How do I find an application cluster resource group (CRG) in iSeries Navigator?
10. How do I find a data cluster resource group (CRG) in iSeries Navigator?
11. I want to be able to see the switchable device (device CRG) status without having to go back up to the Switchable Hardware folder to see it. How can I do this?

Communications

1. What IP address does the Clusters function in iSeries Navigator use to communicate with the nodes in the cluster? Doesn't it use the IP address of the node name?

Security

1. Why are most of the context menus in the Clusters folder in iSeries Navigator disabled or disappeared?
2. Does the Clusters function in iSeries Navigator use Application Administration values?
3. Why does the Clusters function in iSeries Navigator show a signon window to my nodes in the cluster?

Troubleshooting

1. Why isn't the Clusters folder showing up under Management Central?
2. I already have a cluster, but it doesn't show up in the Clusters folder. Why?
3. Why doesn't the latest status show up in the Clusters folder?
4. Why didn't a failover of my switchable hardware group or switchable application occur?
5. I received a message for a damaged object. What can I do about it?
6. I'm using the ″Browse″ button in the wizards for the nodes to browse for IP addresses. Why aren't all of the TCP/IP addresses that I expect showing up in the browse window?
7. Why are most of the context menus in the Clusters folder in iSeries Navigator disabled or disappeared?
8. I was using the ″New Cluster″ wizard and I got a panel titled: ″New Cluster - No Switchable Software Found″. Is this bad?
9. One of my nodes has a status of ″Not communicating″. How do I correct this?

General

Is there a checklist that outlines the prerequisites to creating a cluster?

Yes. Use the Cluster configuration checklist to make sure that you are prepared to configure clusters in your environment.

Back to questions

iSeries Navigator cluster management: Where is the Clusters function located in the iSeries Navigator interface?

The iSeries Navigator cluster management interface is available as a part of the software package IBM iSeries Access. The Clusters function is located in the Management Central folder in iSeries Navigator. See iSeries Navigator cluster management for details.

Back to questions

How do I create a cluster?

To create a simple cluster using the New Cluster wizard in iSeries Navigator, follow these steps:

1. In iSeries Navigator, expand **Management Central**.
2. Right-click **Clusters**, and select **New Cluster**.
3. Follow the wizard's instructions to create a cluster.

For complete details on creating and configuring a cluster, see Configure a cluster.

Back to questions

What is the relationship between the Clusters folder and the Management Central system group?

When you use iSeries Navigator to create a cluster, a system group is also created on the Management Central server. The system group is given the same name as the cluster name and the endpoint systems in the system group are the nodes in the cluster. The system group also has its own special type so that iSeries Navigator knows it is a special system group that represents a cluster.

Important: The Management Central system contains the system groups. If you choose to change your current Management Central system in iSeries Navigator, the new management central system will not have the special cluster system groups, and therefore those clusters will not show up in the Clusters folder.

Back to questions

I already have a cluster defined on some System i products in the network. How do I add it so I can view and manage it through iSeries Navigator?

To add an existing cluster to appear through iSeries Navigator, follow these steps:

1. In iSeries Navigator, expand **Management Central**.
2. Right-click **Clusters**, and select **Add Existing Cluster.**
3. On the **Add Existing Cluster** window, specify one of the servers in the cluster.
4. Click OK.

Back to questions

None of the nodes in my cluster have a status of ″Started″. Which node should I start first?

You should start the node that most recently had a status of ″Started″. For example, say you have two nodes in your cluster: A and B. Node A is currently not started and node B is currently not started. However, node B was the last node to be running with a status of ″Started″. You should start node B first because it will have the most recent information about the cluster.

Back to questions

Why should I care which node is started first?

You should care because the node that most recently had a status of ″Started″ is the node that contains the latest information about the cluster. This is important because if you started the other node that had been down the longest, then it may contain outdated information about the cluster. The danger is that the outdated information can get propagated to the other nodes in the cluster when those other nodes are started. For example, say have a two node cluster with nodes A and B. If node B was the most recently active node with a status of ″Started″, then it will contain the latest cluster information. If you choose to start node A first, then it might contain some outdated information, but will still be started. When you then start node B, it will join with a currently active node in the cluster (it joins with node A). The outdated cluster information from node A will get propagated to node B and the result is that both nodes will contain outdated information about the cluster. This is why it is important to start node B first. The outdated cluster information can have an effect on the configuration of the switchable devices. If you find that you have some problems starting up switchable devices because of disk units reporting in on the backup node when the switchable hardware group is showing a different current node, then you will need to change the role of nodes in the recovery domain, making the node which owns the disk units the primary node.

Back to questions

What does the ″Current Primary Node″ column mean in the Switchable Hardware, Switchable Software, and Switchable Data folders?

The ″Current Primary Node″ column indicates that the node that is currently serving as the primary node for the switchable device or switchable software product. Or, in cluster API terminology, it means that it is the node with the current role in the recovery domain of primary.

Back to questions

How do I find a device cluster resource group (CRG) in iSeries Navigator?

Device CRGs (cluster resource groups) are referred to as Switchable Hardware Groups and found in the **Switchable Hardware** folder in the Clusters folder.

Back to questions

How do I find an application cluster resource group (CRG) in iSeries Navigator?

Application CRGs (cluster resource groups) are referred to as Switchable Software Products and found in the **Switchable Software** folder in the Clusters folder.

Back to questions

How do I find a data cluster resource group (CRG) in iSeries Navigator?

Data CRGs (cluster resource groups) are referred to as Switchable Data Groups and found in the **Switchable Data** folder in the Clusters folder.

Back to questions

I want to be able to see the Switchable Hardware Group (device CRG) status without having to go back up to the Switchable Hardware folder to see it. How can I do this?

As an alternative to navigating to the Switchable Hardware folder every time you want to view status, you can also open a new window with the Switchable Hardware view by right-clicking on the **Switchable Hardware** folder and selecting **Open**. The separate window will show the Switchable Hardware Groups (device CRGs) and their associated status information. This also applies for **Switchable Software** and **Switchable Data**.

Back to questions

Communications: What IP address does the Clusters function in iSeries Navigator use to communicate with the nodes in the cluster? Doesn't it use the IP address of the node name?

There is a ″Server″ column in the main Clusters folder that displays information about your configured clusters. The server name is also on the properties panel for each cluster. The server listed in the ″Server″ column is the node in the cluster that the iSeries Navigator interface uses to communicate with the cluster. It only applies to how iSeries Navigator communicates with the cluster object on the server, not how the nodes in the cluster communicate with one another. The server used by iSeries Navigator cluster management has nothing to do with the current Management Central server.

If the node that iSeries Navigator is using to communicate with the cluster goes down, you can change the communications vehicle to a different node in the cluster to perform cluster actions.

To change the server that will be used by the iSeries Navigator interface to communicate to the cluster, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Clusters**.
3. Right-click the cluster, and select **Change Server**.

Back to questions

Security: Why are most of the context menus in the Clusters folder in iSeries Navigator disabled or disappeared?

Some operations are only available depending on the state of the current configuration of your cluster. For example, you cannot stop a node that is already stopped, you cannot add a node to a cluster that already has the maximum amount of nodes, four, configured. The online help for particular tasks has explanations of why some of these items are disabled or not available.

Some operations are not available if you don't have enough authority. If you are using iSeries Navigator and you have *SECOFR userclass authority, you will have access to all cluster operations and administration. iSeries Navigator uses Application Administration authority from the current Management Central system to determine if you have Application Administration authority for the various iSeries Navigator cluster management operations.

See Application Administration for details on working with Application Administration.

Back to questions

Does the clusters function in iSeries Navigator use Application Administration values?

Yes. iSeries Navigator cluster management uses the Application Administration authority values from the current Management Central system to determine if you have Application Administration authority for various cluster operations.

iSeries Navigator has two types of authority settings for access: **Cluster Operation** and **Cluster Administration**

With the **Cluster Operation** authority, you can:
- View the status of the cluster
- Start and stop nodes
- Start and stop switchable hardware and switchable software
- Perform manual switching of switchable hardware and switchable software

With the **Cluster Administration** authority, you can:
- Create/Delete clusters
- Add and remove nodes

- Add and delete switchable hardware, switchable software, and disk pools
- Change the properties of switchable hardware and switchable software

Back to questions

Why does the Clusters function in iSeries Navigator show a signon window to my nodes in the cluster?

In some cases, iSeries Navigator will try to communicate with all of the nodes in the cluster. This depends on the state of your cluster. When iSeries Navigator needs to communicate with a node, it will first search the existing signon cache in iSeries Navigator to try to find an existing open connection. If it does not find an existing connection, it will then ask the user to sign on. If you cancel the signon window, iSeries Navigator will make an attempt to allow the user to do cluster operations. Some operations may not be possible if iSeries Navigator can′t communicate with the nodes.

Back to questions

Troubleshooting: Why isn′t the Clusters folder showing up under Management Central?

It is possible that you didn′t do a full install of iSeries Access for Windows® on your PC. You may have performed a basic install or chosen some custom options. See iSeries Access for details on installation.

Back to questions

I already have a cluster, but it doesn′t show up in the Clusters folder. Why?

The short answer is this: It isn′t showing up because there is not a system group on your Management Central system that represents the cluster. That system group representing the cluster is created by iSeries Navigator cluster management when either the cluster is created or the cluster is added to the Clusters folder by using the ″add existing cluster″ action. You can expand the **System Groups** folder in Management Central to see the system groups. The cluster system groups will show up as ″third party″ system groups, but don′t assume all ″third party″ system groups are clusters.

Back to questions

Why doesn′t the latest status show up in the Clusters folder?

iSeries Navigator displays information about configured clusters as a shapshot by going out to the cluster nodes and getting the latest information about the cluster and then displaying it in the iSeries Navigator window. It does not automatically perform regular updates of the information. The best way to get the latest snapshot of information is to do a manual refresh. You can use the **View** menu in iSeries Navigator and then choose the **Refresh** option. The alternative is to set up iSeries Navigator to perform automatic refreshes.

Back to questions

Why didn′t a failover of my switchable device, switchable application, or switchable data group occur?

The most likely scenario is that you didn′t have the switchable resource (cluster resource group) started in the cluster. In other words, before the automatic failover was to occur, the status of the switchable resource was not ″Started″. Your switchable resources must be started for a failover to occur.

Back to questions

I received a message for a damaged object. What can I do about it?

You may have received a message like this: `CPF811C User queue QUGCLUSRQ in QCLUMGT damaged`

Option 1: One option is to delete the object and restore it. This is only possible if you previously saved the object.

Option 2: Delete the damaged object. For example, if QUGCLUSRQ in library QCLUMGT is damaged, then delete the object. Then add the existing cluster in iSeries Navigator. By adding the cluster, the cluster GUI will check if the objects exist and re-create them if they don't already exist. See How do I add an existing cluster so I can view and manage it through iSeries Navigator? for details on adding the existing cluster.

Back to questions

I'm using the ″Browse″ button in the wizards for the nodes to browse for IP addresses. Why aren't all of the TCP/IP addresses that I expect showing up in the browse window?

The list is only a candidate list of possible IP addresses. You are not restricted to the list of possible addresses shown in the window. You can enter any cluster interface address you want. Be aware, however, that you will receive errors later if iSeries Navigator can't connect using the IP address you specify as the primary IP address. iSeries Navigator uses the primary IP address to connect to the nodes in the cluster.

Back to questions

I was using the ″New Cluster″ wizard and I got a panel titled: ″New Cluster - No Switchable Software Found″. Is this bad?

No, this is not bad and it is not an error. It means exactly what it says; the iSeries Navigator interface cannot find any switchable software that can be automatically installed using the wizard. iSeries Navigator requires that the any auto-installable switchable software conform to the i5/OS architecture for cluster-enabled applications. Additionally, iSeries Navigator only supports a subset of this architecture, not all of it.

Back to questions

One of my nodes has a status of ″Not communicating″. How do I correct this?

A cluster partition happens if you lose contact between one or more nodes in the cluster and a failure of the lost nodes cannot be confirmed. See Partition errors for more information.

Sometimes a partitioned condition is reported when there really was a node outage. This can occur when cluster resource services loses communications with one or more nodes, but cannot detect if the nodes are still operational. When this condition occurs, a simple mechanism exists for you to indicate that the node has failed. See Change partitioned nodes to failed for details.

Back to questions

Who to call for cluster support

See this topic if you need to contact IBM with your cluster questions.

If you need help in deciding whether clusters can benefit your business, or if you run into problems after you have implement clusters, you can contact the following services:

- For additional technical marketing assistance or if you want to hire IBM consultation services, contact the Continuous Availability Center in the iSeries Technology Center by sending an e-mail to rchclst@us.ibm.com.

- For other problems, contact either the IBM Business Partner you purchased your clustering software package from or call 1-800-IBM-4YOU (1-800-426-4968).

  Related tasks

  "Configuring clusters" on page 110
  Understand how to create a cluster.

Related information for Clusters

Find related information for Clusters.

Redbooks™

- Data Resilience Solutions for IBM i5/OS High Availability Clusters
- Clustering and IASPs for Higher Availability
- High Availability on the AS/400® System: A System Manager's Guide
- IBM eServer iSeries Independent ASPs: A Guide to Moving Applications to IASPs
- The System Administrator's Companion to AS/400 Availability and Recovery

Web sites

- High Availability and Clusters (www.ibm.com/servers/eserver/iseries/ha)
  IBM site for High Availability and Clusters

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As** if you are using Internet Explorer. Click **Save Link As** if you are using Netscape Communicator.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

You need Adobe Acrobat Reader to view or print these PDFs. You can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Programming Interface Information

This Clusters publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

Terms and conditions

Printed in USA