Tivoli® System Automation for Multiplatforms

IBM

**Version 2 Release 2**

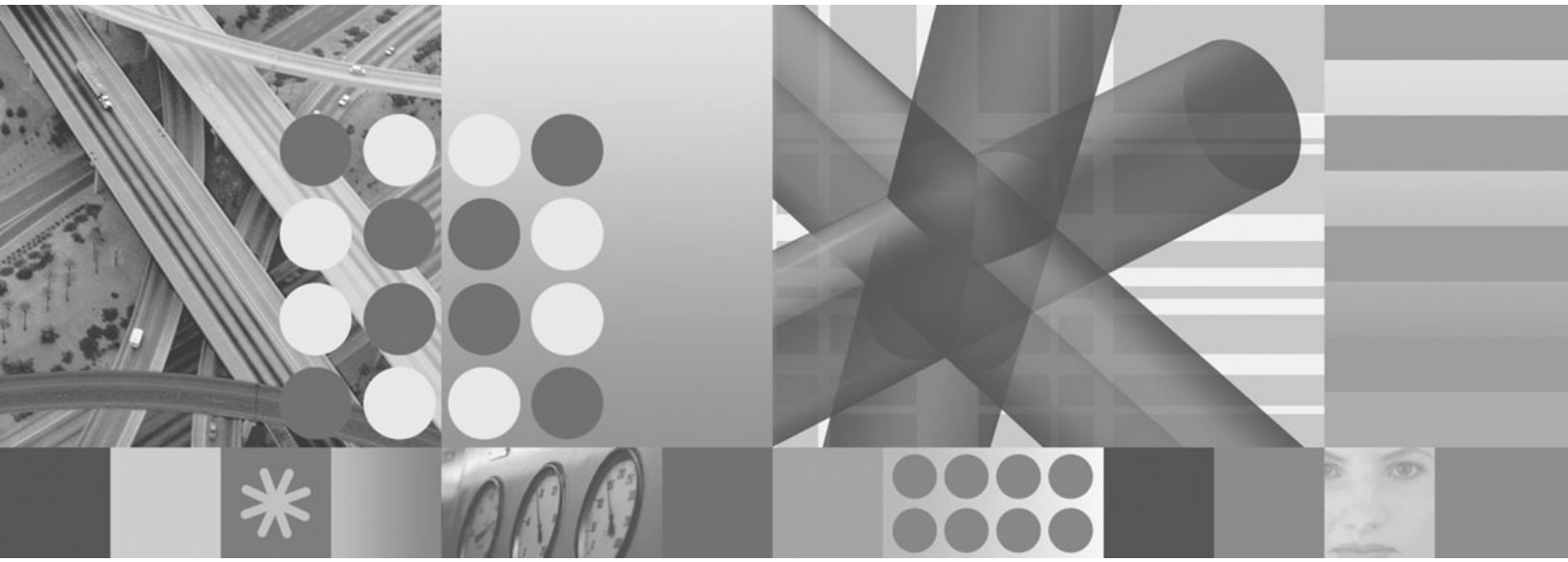
# Base Component
# Administrator's and User's Guide

SC33-8272-00

Tivoli® System Automation for Multiplatforms

IBM

**Version 2 Release 2**



# Base Component
# Administrator's and User's Guide

SC33-8272-00

**Note!**
Before using this information and the product it supports, read the information in Appendix B, "Notices," on page 249.

**First Edition (October 2006)**

This edition of the *IBM Tivoli System Automation for Multiplatforms Base Component Administrator's and User's Guide* applies to Version 2, Release 2, Modification 0 of IBM Tivoli System Automation for Multiplatforms, program number 5724–M00, and to all subsequent releases and modifications of this product until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Str. 220
D-71032 Boeblingen
Federal Republic of Germany

FAX (Germany): 07031+16-3456
FAX (Other Countries): (+49)+7031-16-3456

Internet e-mail: eservdoc@de.ibm.com

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

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

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

# Contents

# Figures

# Tables

# About this book

This book provides the information needed to implement the IBM Tivoli System Automation for Multiplatforms policy-based self-healing capability running on IBM System x, System z, System i, System p, and AIX®.

## Who should use this book

This book is intended for system administrators and operators who want to use the automation and failover capabilities of the **Base component** of IBM Tivoli System Automation for Multiplatforms.

## How to use this book

This book contains the information you need to understand and use the Base component of IBM Tivoli System Automation for Multiplatforms (IBM Tivoli System Automation).

- Chapter 1 provides an introduction to IBM Tivoli System Automation. It gives an overview of IBM Tivoli System Automation, introduces the components, and explains the technical terms used in this manual.
- Chapter 2 is a quick guide to the tasks you need to complete to work with the base component. It describes how clusters and nodes are administered and how an automation policy is defined.
- Chapter 3 describes the common attributes of IBM Tivoli System Automation.
- Chapter 4 describes how resource groups are created and used.
- Chapter 5 describes how equivalencies are created and used.
- Chapter 6 describes how managed relationships are created and used.
- Chapter 7 describes how IBM Tivoli System Automation processes system information.
- Chapter 8 describes the usage of the operations console.
- Chapter 9 describes how IBM Tivoli System Automation protects your resources.
- Chapter 10 describes how a highly available network is set up.
- Chapter 11 describes how IBM Tivoli System Automation is controlled and administered.
- Chapter 12 describes the resource managers provided by IBM Tivoli System Automation.
- The appendix provides useful information for troubleshooting.

## What's new in release 2.2

**New commands**

In release 2.2, a number of commands are introduced that provide improvements in usability and reliability for the base component of IBM Tivoli System Automation for Multiplatforms:

**samsimul command**

Simulates a series of requests against or state changes of individual automated resources and displays the expected consequences. The automation engine simulates all actions that are triggered by the requests or state changes rather than performing them. Developers

of system automation policies can use the **samsimul** command to test the results that the activation of a policy would have in arbitrary variations before actually activating it.

**lssam command**
Lists resource groups, nested groups, and contained members together with their operational state in a tree format. This allows operators to quickly get a clear overview of all currently defined resources and their states.

**prereqSAM command**
Checks whether all prerequisites for installing the base component are met. It is invoked implicitly by the **installSAM** command, but can also be invoked separately by a user prior to starting the installation. **prereqSAM** reports if any prerequisites are missing or not at the required minimum level.

**Changes to the documentation**
The *IBM Tivoli System Automation for Multiplatforms Base Component User's Guide* has been renamed to *IBM Tivoli System Automation for Multiplatforms Base Component Administrator's and User's Guide*.

The installation and configuration of the base component, end-to-end automation adapter, and base component operations console, which were formerly described in the Base Component User's Guide, are now described in the new *IBM Tivoli System Automation for Multiplatforms Installation and Configuration Guide*.

The manuals that make up the IBM Tivoli System Automation for Multiplatforms library are listed in "Where to find more information."

## Where to find more information

In addition to this manual, the IBM Tivoli System Automation for Multiplatforms library comprises the following books:

- *IBM Tivoli System Automation for Multiplatforms Installation and Configuration Guide*, SC33-8273
- *IBM Tivoli System Automation for Multiplatforms Base Component Reference*, SC33-8274
- *IBM Tivoli System Automation for Multiplatforms End-to-End Automation Management Component Administrator's and User's Guide*, SC33-8275
- *IBM Tivoli System Automation for Multiplatforms End-to-End Automation Management Component Reference*, SC33-8276

You can download the complete documentation at
`http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html`

The IBM Tivoli System Automation home page offers up-to-date information and services, and other items of interest to IBM Tivoli System Automation users.

You find the IBM Tivoli System Automation home page at
`www.ibm.com/software/tivoli/products/sys-auto-linux`

## Conventions

The following highlighting conventions are used in this book:

| | |
|---|---|
| **Bold** | Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects. |
| *Italic* | Identifies parameters whose actual names or values are to be supplied by the user. |
| `monospace` | Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type. |

This manual uses symbols to show resources, resource groups, equivalencies, and relationships. The symbols used are as follows:

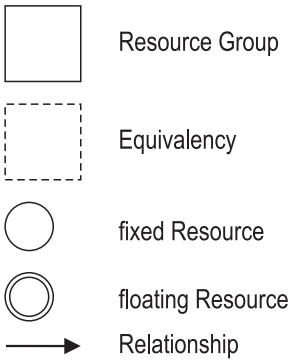



*Figure 1. Symbols used in this guide*

## ISO 9000

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

## Related information

The following IBM Reliable Scalable Cluster Technology (RSCT) publications are available on the IBM Tivoli System Automation Base Component CD:

- *RSCT Administration Guide*, SA22-7889.
- *RSCT for AIX 5L: Technical Reference*, SA22-7890.
- *RSCT for Linux: Technical Reference*, SA22-7893.
- *RSCT Messages*, GA22-7891.
- *RSCT Diagnosis Guide*, SA23-2202.

RSCT publications can also be found at the following Web site:

`www.ibm.com/servers/eserver/clusters/library/`

You might also need to refer to the following IBM Redpaper:

- *Linux on IBM zSeries® and S/390®: High Availability for z/VM® and Linux*

It can be found at the following Web site:

http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp0220.html

## How to obtain publications

The IBM Tivoli System Automation publications are also available (valid at the time of release) at these Web sites:

www.ibm.com/servers/eserver/clusters/library/
www.ibm.com/servers/eserver/zseries/software/sa/
www.ibm.com/software/sysmgmt/products/support/

## How to reach us by e-mail

If you would like to contact us by e-mail, send your comments to eservdoc@de.ibm.com

# Chapter 1. Introduction

## Overview

IBM Tivoli System Automation manages the availability of applications running in Linux systems or clusters on xSeries, zSeries, iSeries, pSeries, and AIX systems or clusters. It consists of the following features:

### High availability and resource monitoring

IBM Tivoli System Automation provides a high availability environment. High availability describes a system which is continuously available and which has a self-healing infrastructure to prevent downtime caused by system problems. A self-healing infrastructure detects improper operation of system, transactions and processes, and initiates corrective action without disrupting users. IBM Tivoli System Automation offers mainframe-like high availability by using fast detection of outages and sophisticated knowledge about application components and their relationships. It provides quick and consistent recovery of failed resources and whole applications either in place or on another system of a Linux cluster or AIX cluster without any operator intervention. Thus it relieves operators from manual monitoring, remembering application components and relationships, and therefore eliminates operator errors.

### Policy based automation

IBM Tivoli System Automation allows to configure high availability systems through the use of policies that define the relationships among the various components. These policies can be applied to existing applications with minor modifications. Once the relationships are established, IBM Tivoli System Automation will assume responsibility for managing the applications on the specified nodes as configured. This reduces implementation time and the need for complex coding of applications. In addition, systems can be added without modifying scripts, and resources can be easily added, too.
There are sample policies available for IBM Tivoli System Automation. You can download them from the IBM Support FTP site at:

`ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux/`

### Automatic recovery

IBM Tivoli System Automation quickly and consistently performs an automatic restart of failed resources or whole applications either in place or on another system of a Linux or AIX cluster. This greatly reduces system outages.

### Automatic movement of applications

IBM Tivoli System Automation manages the cluster-wide relationships among resources for which it is responsible. If applications need to be moved among nodes, the start and stop relationships, node requirements and any preliminary or follow-up actions are automatically handled by IBM Tivoli System Automation. This again relieves the operator from manual command entry, reducing operator errors.

## Resource grouping

Resources can be grouped together in IBM Tivoli System Automation. Once grouped, all relationships among the members of the group can be established, such as location relationships, start and stop relationships, and so on. After all configuration is completed, operations can be performed against the entire group as a single entity. This once again eliminates the need for operators to remember the application components and relationships, reducing the possibility of errors.

# IBM Tivoli System Automation terms

This section gives an overview of the terms this manual uses when describing IBM Tivoli System Automation.

## Cluster / peer domain

The group of host systems upon which IBM Tivoli System Automation manages resources is known as a cluster. A cluster can consist of one or more systems or nodes. Throughout this manual 'peer domain' is also used when referring to a cluster. The two terms are interchangeable. IBM Tivoli System Automation supports up to 32 nodes within a cluster.

## Resource

A resource is any piece of hardware or software that can be defined to IBM Tivoli System Automation. These resources can be either defined manually by the administrator using the mkrsrc (make resource) command or through the 'harvesting' functionality of the cluster infrastructure, whereby resources are automatically detected and prepared for use. All resources are controlled through the appropriate resource managers as described in "Introducing resources managers provided by IBM Tivoli System Automation" on page 6. Resources have characteristics, or attributes, which can be defined. For example, when considering an IP address as a resource, attributes would include the IP address itself and the net mask. There are two types of resources: fixed resources and floating resources.

**Fixed resource**
A fixed resource is a resource that has only a single instance within the cluster. It represents one entity that is defined for a single node, and this is the only node on which it runs.

**Floating resource**
A floating resource is a resource which can run on several nodes in the cluster. You can find a detailed definition of a floating resource under "ResourceType attribute" on page 27.

## Resource attributes

A resource attribute describes some characteristics of a resource. There are two types of resource attributes: persistent attributes and dynamic attributes.

**Persistent attributes**
The attributes of the IP address just mentioned (the IP address itself and the net mask) are examples of persistent attributes – they describe enduring characteristics of a resource. While you could change the IP address and net mask, these characteristics are, in general, stable and unchanging.

**Dynamic attributes**
Dynamic attributes, on the other hand, represent changing characteristics of the resource. Dynamic attributes of an IP address, for example, would identify such things as its operational state.

## Resource class

A resource class is a collection of resources of the same type. For example, if an application is a resource, then all applications defined in the cluster would comprise a resource class. Resource classes allow you to define the common characteristics among the resources in its class. In the case of applications, the resource class can define identifying characteristics, such as the name of the

application, and varying characteristics, such as whether or not the application is running. So each resource in the class can then be noted by its characteristics at any given time. Resource classes are managed by various resource managers – see "Introducing resources managers provided by IBM Tivoli System Automation" on page 6.

## Resource group

Resource groups are logical containers for a collection of resources. This container allows you to control multiple resources as a single logical entity. Resource groups are the primary mechanism for operations within IBM Tivoli System Automation. Resource groups can also be nested, meaning that applications can be split into several resource groups which themselves are part of another higher level resource group. Also resource groups can be defined in such a way that their members can be located on different systems in the cluster.

## Managed resource

A managed resource is a resource that has been defined to IBM Tivoli System Automation. To accomplish this, the resource is added to a resource group, at which time it becomes manageable through IBM Tivoli System Automation.

## Nominal state

The nominal state of a resource group indicates to IBM Tivoli System Automation whether the resources with the group should be Online or Offline at this point in time. So setting the nominal state to ″Offline″ indicates that you wish for IBM Tivoli System Automation to stop the resources in the group, and setting the nominal state to ″Online″ is an indication you wish to start the resources in the resource group. You can change the value of the NominalState resource group attribute, but you cannot set the nominal state of a resource directly. See "NominalState attribute" on page 37.

## Equivalency

An equivalency is a collection of resources that provides the same functionality. For example, equivalencies are used for selecting network adapters that should host an IP address. If one network adapter goes offline, IBM Tivoli System Automation selects another network adapter to host the IP address.

## Relationships

IBM Tivoli System Automation allows the definition of relationships between resources in a cluster. There are two different relationship types:

- **Start-/stop relationships**

  Relationships are used to define start and stop dependencies between resources. You can use the StartAfter, StopAfter, DependsOn, DependsOnAny, and ForcedDownBy relationships to achieve this. For example, a resource must only be started after another resource was started. You can define this by using the policy element StartAfter relationship.

- **Location relationships**

  Location relationships are applied when resources must, or should if possible, be started on the same or a different node in the cluster. IBM Tivoli System Automation provides the following location relationships: Collocation, AntiCollocation, Affinity, AntiAffinity, and IsStartable. A simple example is that a webserver and its corresponding service IP address, which could be started on any node in the cluster, should always be kept together. In the past this behavior

had to be defined by writing complex scripts. Now IBM Tivoli System Automation allows the usage of a location relationship which simplifies the policy definition for the administrator.

Relationships provide the following additional features:

- The possibility to define relationships between resource groups, resources, and equivalencies.
- The possibility to define relationships between resources running on different systems in the cluster.

## Quorum

The main goal of quorum operations is to keep data consistent and to protect critical resources. Quorum can be seen as the number of nodes in a cluster that are required to modify the cluster definition or perform certain cluster operations. There are two types of quorum:

**Configuration quorum**

This quorum determines when configuration changes in the cluster will be accepted. Operations affecting the configuration of the cluster or resources are only allowed when the absolute majority of nodes is online. See "Configuration quorum" on page 138 for a detailed description.

**Operational quorum**

This quorum is used to decide whether resources can be safely activated without creating conflicts with other resources. In case of a cluster splitting resources can only be started in the subcluster which has a majority of nodes or obtained a tie breaker. See "Operational quorum" on page 138 for a detailed description.

## Tie breaker

In case of a tie in which a cluster has been partitioned into subcluster with an equal number of nodes, the tie breaker is used to determine which subcluster will have an operational quorum.

# Components of IBM Tivoli System Automation

Reliable Scalable Cluster Technology, or RSCT, is a product fully integrated into IBM Tivoli System Automation. RSCT is a set of software products that together provide a comprehensive clustering environment for AIX and Linux. RSCT is the infrastructure to provide clusters with improved system availability, scalability, and ease of use.

RSCT provides three basic components, or layers, of functionality:

- **RMC** (Resource Monitoring and Control), provides global access for configuring, monitoring, and controlling resources in a peer domain.
- **HAGS** (High Availability Group Services), is a distributed coordination, messaging, and synchronization service.
- **HATS** (High Availability Topology Services), provides a scalable heartbeat for adapter and node failure detection, and a reliable messaging service in a peer domain.

## Introducing resources managers provided by IBM Tivoli System Automation

Resource classes are managed by the various resource managers (RM), depending on what type of resource is being managed. A resource manager is a software layer between a resource and RMC.
The following resource managers are provided by IBM Tivoli System Automation:

### Recovery RM (IBM.RecoveryRM)

This resource manager serves as the decision engine for IBM Tivoli System Automation. Once a policy for defining resource availabilities and relationships is defined, this information is supplied to the Recovery RM. This RM runs on every node in the cluster, with exactly one Recovery RM designated as the master. The master evaluates the monitoring information from the various resource managers. Once a situation develops that requires intervention, the Recovery RM drives the decisions that result in start or stop operations on the resources as needed.

### Global Resource RM

The Global Resource RM (IBM.GblResRM) supports two resource classes:

**IBM.Application**
: The IBM.Application resource class defines the behavior for general application resources. This class can be used to start, stop, and monitor processes. As a generic class, it is very flexible and can be used to monitor and control various kind of resources. Most of the applications that you will automate will be done using this class. For more information, refer to “Using the Global Resource Manager” on page 197.

**IBM.ServiceIP**
: This application class defines the behavior of Internet Protocol (IP) address resources. It allows you to assign IP addresses to an adapter. In effect, it allows IP addresses to ’float’ among nodes. For more information, refer to “What is the IBM.ServiceIP resource class?” on page 211.

### Configuration RM

The Configuration RM (IBM.ConfigRM) is used in cluster definition. In addition, quorum support, which is a means of insuring data integrity when portions of a cluster lose communication, is provided.

### Event response RM

The Event Response RM (IBM.ERRM) provides the ability to monitor conditions in the cluster in order for the RMC system to react in certain ways.

### Test RM

The Test resource manager (IBM.TestRM) manages test resources and provides functions to manipulate the operational state of these resources. The resource manager is operational in a peer domain mode only and provides the resource class IBM.Test. The Test resource manager does not control real resources. A detailed description of the Test RM is given in Chapter 12, "Resources managers provided by IBM Tivoli System Automation," on page 197.

Figure 2 shows a diagram of the previously described components.

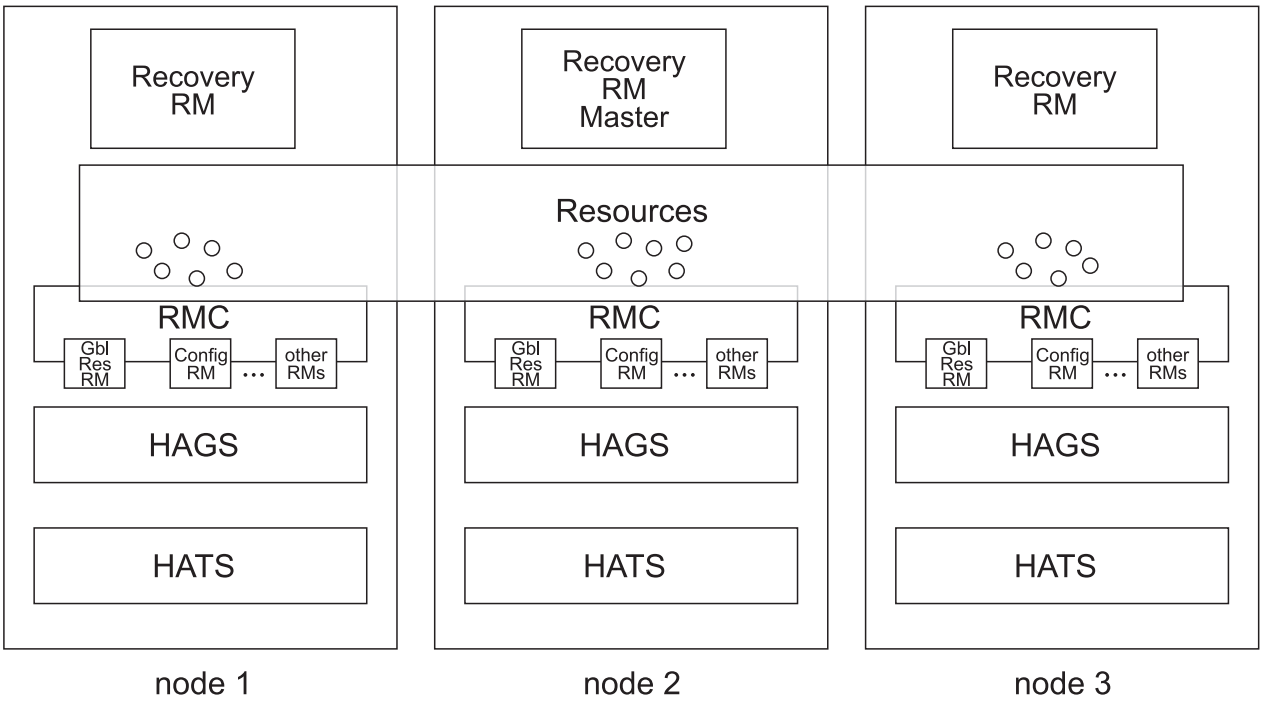



*Figure 2. Overview of IBM Tivoli System Automation architecture*

## Operations console

The operations console is a browser-based graphical user interface. The operations console is used to monitor and manage resources managed by IBM Tivoli System Automation. This is done in a mode called direct access mode.

For the end-to-end automation management component of IBM Tivoli System Automation for Multiplatforms two more modes are provided:

- End-to-end automation mode.
- First-level automation mode.

For more information on these modes, see the *IBM Tivoli System Automation for Multiplatforms End-to-End Automation Management Component Administrator's and User's Guide*.

## End-to-end automation adapter

In the base component of IBM Tivoli System Automation the end-to-end automation adapter is used to operate automated resources directly from the operations console. For more information on the end-to-end automation adapter, see the *IBM Tivoli System Automation for Multiplatforms Installation and Configuration Guide*.

In the end-to-end automation management component of IBM Tivoli System Automation the end-to-end automation adapter is used to automate the operation of resources within first-level automation domains.

# Chapter 2. Getting started

This chapter lists and describes the steps shown below which you have to perform to start IBM Tivoli System Automation:

**Step 1: Defining and administering a cluster (see page 10)**
This step shows how you can create and remove a cluster, how you can add nodes to a cluster and remove nodes from a cluster, and how you can check the status of the IBM Tivoli System Automation daemon.

**Step 2: Defining RSCT resources (see page 16)**
This step shows how you can create a resource like a web server and how you can create an equivalency relationship.

**Step 3: Defining the automation policy (see page 20)**
This step shows you can define the relationships among the components created in Step 1 and Step 2. This is called defining the automation policy.

For detailed information about the commands that are mentioned in this chapter, refer to the following documentation:

**RSCT commands**
For a detailed description of the RSCT commands refer to the appropriate man pages and to the RSCT manuals listed below. Note that the following RSCT commands are only described in the man pages:
**resetrsrc**, **startrsrc**, **stoprsrc**

The RSCT manuals are available on the IBM Tivoli System Automation CD. They can also be downloaded from the following Web site:

```
www.ibm.com/servers/eserver/clusters/library/
```

The RSCT commands are described in the following IBM Reliable Scalable Cluster Technology (RSCT) manuals:

- *RSCT Administration Guide*, SA22-7889
- *RSCT for AIX 5L: Technical Reference*, SA22-7890
- *RSCT for Linux: Technical Reference*, SA22-7893

**IBM Tivoli System Automation commands**
For a detailed description of these commands, refer to the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*. You will use some of these commands in Step 3.

# Step 1: Defining and administering a cluster

The scenarios described below show how you can create a cluster, set up a tie breaker, add nodes to the cluster, and how you can check the status of the IBM Tivoli System Automation daemon (IBM.RecoveryRM).

## Command overview

The following gives you an overview of the *Reliable Scalable Cluster Technology (RSCT) for Linux* commands you will use when working with cluster definitions. You will need some of these commands when you are going through Step 1. For a detailed description of the commands, refer to the RSCT manuals listed on page 9.

**preprpnode** This command prepares the security settings for the node to be included in a cluster. When issued, public keys are exchanged among the nodes, and the RMC access control list (ACL) is modified to enable access to cluster resources by all the nodes of the cluster.

**mkrpdomain** This command creates a new cluster definition. It is used to specify the name of the cluster, and the list of nodes to be added to the cluster.

**lsrpdomain** This command lists information about the cluster to which the node where the command runs belongs.

**startrpdomain / stoprpdomain**
These commands are used to bring the cluster online and offline, respectively.

**addrpnode** Once a cluster has been defined and is operational, this command is used to add new nodes to the cluster.

**startrpnode / stoprpnode**
These commands are used to bring individual nodes online and offline to the cluster. They often used when performing maintenance to a particular system. The node is stopped, repairs or maintenance is performed, then the node is restarted, at which time it rejoins the cluster.

**lsrpnode** This command is used to view the list of nodes defined to a cluster, as well as the operating state (OpState) of each node. Note that this command is useful only on nodes that are Online in the cluster, otherwise it will not display the list of nodes.

**lssrc** This command lists the status of subsystems.

**rmrpdomain** This command removes a defined cluster.

**rmrpnode** This command removes one or more nodes from a cluster definition.

**startsrc** This command starts an individual subsystem.

**stopsrc** This command starts an individual subsystem.

## Creating a two node cluster

Before beginning to create a cluster you should ensure that your network setup is correct:

- IP, netmask and broadcast addresses must be consistent on each cluster node.
- Make sure that the name resolution is correct, DNS entries are consistent or entries in your local */etc/hosts* files on all nodes are identical.
- Do not define more than one network interface on a node to the same subnet.
- All firewalls, including local firewalls, must allow ICMP pings and IP-traffic to flow through from *12347/udp* for *cthats*, *12348/udp* for *cthags*, *rmc 657/udp* and *tcp*.

For more information, refer to Chapter 10, "Setting up a highly available network," on page 155.

To create a two node cluster, you need to:

1. Access a console on each node in the cluster and log in as root.
2. Set the environment variable CT_MANAGEMENT_SCOPE=2 on each node.
3. Remove extraneous entries from the `/etc/hosts` table that refer to the same host name (Example: 127.0.0.2 my_hostname). These entries are created during the installation of the operating system but conflict with the cluster infrastructure.
4. Issue the **preprpnode** command on all nodes to allow communication between the cluster nodes.

   ```
   preprpnode node01 node02
   ```

5. You can now create a cluster with the name **SA_Domain** running on node01 and node02. The following command can be issued from any node.

   ```
   mkrpdomain SA_Domain node01 node02
   ```

   Note that when creating RSCT peer domains (clusters) using **mkrpdomain**, the characters used for the peer domain name are limited to the following ASCII characters: A-Z, a–z, 0-9, . (period), and _(underscore).
6. To look up the status of **SA_Domain**, issue the **lsrpdomain** command:

   ```
   lsrpdomain
   ```

   Output:

   ```
   Name       OpState RSCTActiveVersion MixedVersions TSPort GSPort
   SA_Domain  Offline 2.4.4.2           No            12347  12348
   ```

   The cluster is defined but offline.
7. Issue the **startrpdomain** command to bring the cluster online.

   ```
   startrpdomain SA_Domain
   ```

   When you run the **lsrpdomain** command again, you see that the cluster is still in the process of starting up, the OpState is Pending Online.

   ```
   Name       OpState        RSCTActiveVersion MixedVersions TSPort GSPort
   SA_Domain  Pending online 2.4.4.2           No            12347  12348
   ```

   After a short time the cluster will be started, so when issuing the **lsrpdomain** again, you see that the cluster is now online:

   ```
   Name       OpState RSCTActiveVersion MixedVersions TSPort GSPort
   SA_Domain  Online  2.4.4.2           No            12347  12348
   ```

**Notes:**

1. You may get an error message like:

```
2632-044 The domain cannot be created due to the following errors that were
detected while harvesting information from the target nodes:
node1: 2632-068 This node has the same internal identifier as node2 and
cannot be included in the domain definition.
```

This error most often occurs if you have cloned Linux images.

Something went wrong with the cluster and the entire configuration should be reset. Solve such problems by running the

```
/usr/sbin/rsct/install/bin/recfgct
```

command on the node which is named in the error message in order to reset the node id. Continue with the **preprpnode** command.

2. You may also get an error message like:

```
2632-044 The domain cannot be created due to the following errors
that were detected while harvesting information from the target nodes:
node1: 2610-418 Permission is denied to access the resources
or resource class specified in  this command.
```

Check your hostname resolution. Make sure that all entries for each node of the cluster in your local */etc/hosts* files on all nodes and the nameserver entries are identical.

3. You may get an error message like:

```
2612-022 A session could not be established with the RMC daemon on <node_name>.
```

This error most often occurs if a firewall is running on one of the cluster nodes. To resolve the problem enable the following ports in the firewall software:

```
rmc port 657 TCP IN/OUT Source Port Range: ephemeral port
rmc port 657 UDP IN/OUT Source Port Range: ephemeral port
cthats port 12347 UDP IN/OUT Source Port Range: 1024 - 65535
cthags port 12348 UDP IN/OUT Source Port Range: 1024 - 65535
```

Note that all firewall rules should allow BROADCAST packets to go through the cthats port. For more information, refer to *IBM RSCT Administration Guide*, SA22-7889 (Appendix A. RSCT network considerations).

## Setting up a tie breaker

Tie breakers are only needed for clusters that consist of an even number of nodes.

For two node clusters, a tie breaker is required for the following reasons:

- Without a tie breaker, IBM Tivoli System Automation automates resources only when more than half of the nodes of a cluster are online. For a two node cluster without a tie breaker this means that both nodes must be online for automation to be performed.
- A tie breaker ensures that the cluster remains operational even when it is split in such a way that the communication between the two nodes fails.

For all other clusters with an even number of nodes, a tie breaker should be set up. The tie breaker ensures that situations can be resolved in which only half of the nodes of the cluster are online or when the cluster is split.

For detailed information about setting up, administering, and using tie breakers, refer to Chapter 9, “Protecting your resources – quorum support,” on page 137.

## Adding a node to an existing cluster

After having created a two node cluster, you might want to add a third node to **SA_Domain**. In order to do this, you need to:

1. Issue the **lsrpdomain** command to see if your cluster is online:

```
Name       OpState RSCTActiveVersion MixedVersions TSPort GSPort
SA_Domain  Online  2.4.4.2           No            12347  12348
```

   Issue the **lsrpnode** command to see which nodes are online:

```
Name    OpState   RSCT Version
node02  Online    2.4.4.2
node01  Online    2.4.4.2
```

2. Issue the following **preprpnode** commands to allow communication between the existing nodes and the new node.
   Log on to node03 and enter:

```
preprpnode node01 node02
```

   Log on to node02 and enter:

```
preprpnode node03
```

   Log on to node01 and enter:

```
preprpnode node03
```

   You are strongly recommended to issue a preprpnode command on each node for all nodes.

3. In order to add the node03 to the cluster definition, issue the **addrpnode** command on node01 or node02, which are already online on the cluster.

```
addrpnode node03
```

   Again issue the **lsrpnode** command to see the status of all nodes:

```
Name    OpState   RSCT Version
node02  Online    2.4.4.2
node03  Offline   2.4.4.2
node01  Online    2.4.4.2
```

4. Start node03 from an online node:

```
startrpnode node03
```

   After a short time node03 should be online, too.

## Taking an entire cluster or individual nodes offline

In order to perform node maintenance or make application upgrades, you might want to take an entire cluster or individual nodes of a cluster offline:

- In order to perform maintenance on cluster **SA-Domain**, you might wish to take it offline. Use the **stoprpdomain** command from any online node in the cluster to do this.

```
stoprpdomain SA_Domain
```

  Issue the **lsrpdomain** command to check the status of cluster **SA-Domain**:

```
Name       OpState  RSCTActiveVersion MixedVersions TSPort GSPort
SA_Domain  Offline  2.4.4.2           No            12347  12348
```

  Stopping a cluster does not remove the cluster definition: the cluster can therefore be brought back online using the **startrpdomain** command.

- To take one or more cluster *nodes* offline, you use the **stoprpnode** command. You might need to do this to perform application upgrades, to perform maintenance on a node, or before removing the node from the cluster. Also, since a node may be defined in multiple clusters, but online in only one cluster at a time, you might need to take a node offline in one cluster so that you may bring it online

in another cluster. To take a node offline, issue the **stoprpnode** command from any online node in the cluster, and pass to it the cluster node name of the node to take offline. For example, do this to stop node03:

```
stoprpnode node03
```

**Note:** Be careful when stopping multiple nodes of a cluster. You will lose quorum if less than half of the nodes are online. This may lead to outages if there are resources running on the online nodes of the cluster. See Chapter 9, "Protecting your resources – quorum support," on page 137 for more information.

Issue the **lsrpnode** command to see if node03 has gone offline:

```
lsrpnode node03
```

```
Name    OpState   RSCT Version
node03  Offline   2.4.4.2
```

## Removing nodes from a cluster, or removing a complete cluster

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

- To remove a node from a cluster, you use the **rmrpnode** command. In order to remove a node from a cluster, the node must be offline. If the node you wish to remove is not currently offline, you must use the **stoprpnode** command to take it offline. You can also remove *multiple* nodes from the cluster, using the **rmrpnode** command. In order to see which node is offline, issue the **lsrpnode** command from any online node in the cluster.

  ```
  lsrpnode
  ```

  ```
  Name    OpState   RSCT Version
  node02  Online    2.4.4.2
  node03  Offline   2.4.4.2
  node01  Online    2.4.4.2
  ```

  Then issue the rmrpnode command from any online node in the cluster to remove node03.

  ```
  rmrpnode node03
  ```

  Issue the lsrpnode command again to see if node03 has been removed.

  ```
  lsrpnode
  ```

  ```
  Name    OpState   RSCT Version
  node02  Online    2.4.4.2
  node01  Online    2.4.4.2
  ```

- To remove a complete cluster definition, you use the **rmrpdomain** command. Removing a cluster involves removing the cluster definition from each node on the cluster. To do this efficiently, all nodes in the cluster should be online. You can bring individual nodes online using the **startrpnode** command, or you can bring *all* offline nodes in the cluster online using the **startrpdomain** command. The **rmrpdomain** command removes the cluster definition on all of the nodes that are reachable from the node where the command was issued. If the command is issued from an online node in a cluster, and all the nodes are online, then the command will attempt to remove all of their cluster definition files. If a node is not reachable from the node where the **rmrpdomain** is run (for example, the node is offline or inoperative), the **rmrpdomain** command will not be able to remove the cluster definition on that node. In case the cluster cannot

be brought online, you can use the force option *-f* to remove nodes or the cluster. Issue the **startrpdomain** command to bring all nodes of cluster **SA_Domain** online:

```
startrpdomain SA_Domain
```

Then issue the rmrpdomain command to remove cluster **SA_Domain**:

```
rmrpdomain SA_Domain
```

## Administering the recovery resource manager

On each online node in the cluster an IBM Tivoli System Automation daemon (IBM.RecoveryRM) is running. You can check the status and the process id of the daemon with the command lssrc:

```
lssrc -s IBM.RecoveryRM
```

You get the following output:

```
Subsystem         Group            PID     Status
 IBM.RecoveryRM   rsct_rm          18283   active
```

This daemon runs on each node which is online in the cluster. It is started automatically if the cluster node starts. If necessary, you can manually stop the daemon with the command:

```
stopsrc -s IBM.RecoveryRM
```

To start the daemon you use:

```
startsrc -s IBM.RecoveryRM
```

One of the daemons is the so called 'master daemon'. This daemon is responsible for driving all the necessary decisions. You can find out the node the master daemon is located on with the command:

```
lssrc -ls IBM.RecoveryRM | grep Master
```

and you get the following output:

```
   Master Node Name     : node03 (node number = 3)
```

In the example the master daemon runs on a node called node03.

The other daemons are called 'peer daemons'. These peer daemons are a hot standby if the master daemon or the node the master daemon is located on runs into problems. In this case, one of the peer daemons becomes the master. Of course, the takeover between the daemons is done without interruption of the automation functionality of IBM Tivoli System Automation.

## Step 2: Defining RSCT resources

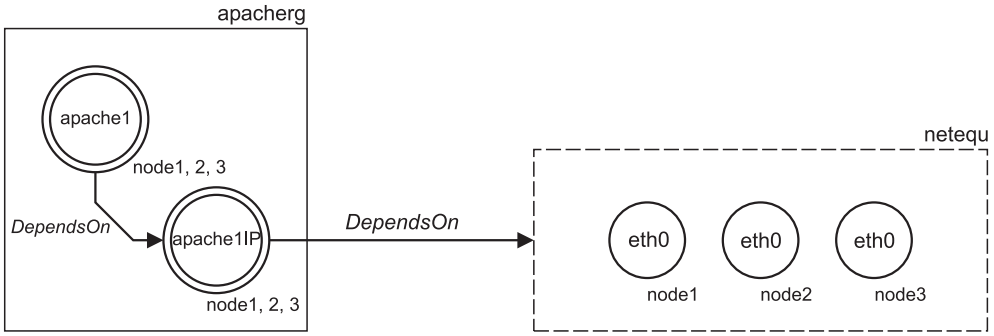
The following example shows how to define a high-available web server on the three nodes of the cluster SA_Domain. See “Step 1: Defining and administering a cluster” on page 10 how this cluster and the nodes node01, node02, and node03 were defined.

The following requirements have to be met for the high-available web server::

- The web server should be startable on any node in the cluster, but will only run on one node at any point in time.
- The web server should be restarted automatically on the same or another node in the cluster in case of a failure. This mechanism also allows a planned outage of nodes for service and maintenance.
- The web server should be addressable with the same IP address regardless of the node it currently runs on. Thus the location of the web server is transparent outside the cluster where no adaption has to be performed, when the web server is moved from one node to another.

As the base for automation, the components involved must first be described in a set of RSCT defined resources. Due to often uncommon characteristics of resources, there are various RSCT resource classes to accommodate the differences. In this example we will need to define three RSCT resources from different classes:

1. An application resource named **apache1**, which represents the web server daemon. The resource is from a class called IBM.Application.
   apache1 will be a floating resource, since the web server is not tied to a specific node in the cluster.
2. An IP address named **apache1IP**, which is used to represent the web server’s IP address. apache1IP is from a class called IBM.ServiceIP. apache1IP will be a floating resource, since it can move around in the cluster following the location of the web server.
3. An representation named **netequ** for the network interface cards that can be used for the apache1IP address. This is called an equivalency and belongs to a class IBM.Equivalency. The characteristic ”floating” or “fixed” is not meaningful for this class.



## Command overview

The following gives you an overview of the *Reliable Scalable Cluster Technology (RSCT) for Linux* commands you will use when defining RSCT resources. You will need some of these commands when you are going through Step 2. For a detailed description of the commands, refer to the RSCT manuals listed on page 9. Note that a description of the RSCT command **resetrsrc** is only available on the corresponding man page.

**chrsrc** This command changes persistent attribute values of a resource within a specified resource class.

**lsrsrc** This command lists resources of a resource class.

**mkrsrc**
This command creates resources of a specified resource class.

**resetrsrc**
This command resets resources of a specified resource class.

**rmrsrc** This command removes resources of a specified resource class.

**runact** This command runs an action on a resource class.

**startrsrc**
This command brings an individual resource online.

**stoprsrc**
This command brings an individual resource offline.

## Creating application resource apache1

As part of the definition of application resource apache1, commands or scripts for starting, stopping and querying the web server have to be specified. These commands and/or scripts can be different ones, but it is often convenient to gather these functions in a single script, which has a command line parameter to select start/stop/status actions. These scripts will often be user-written. See Chapter 12, “Resource managers provided by IBM Tivoli System Automation,” on page 197 for details on the requirements of such scripts.

In this example, we use a script

```
/cluster/scripts/apache
```

which has the following content for a Linux system:

```
#!/bin/bash

OPSTATE_ONLINE=1
OPSTATE_OFFLINE=2

Action=${1}

case ${Action} in
        start)
          /usr/sbin/apachectl start >/dev/null 2>&1
          logger -i -t "SAM-apache" "Apache started"
          RC=0
          ;;
 stop)
          /usr/sbin/apachectl stop >/dev/null 2>&1
          logger -i -t "SAM-apache" "Apache stopped"
          RC=0
          ;;
 status)
          ps -ax |grep -v "grep"|grep "/usr/sbin/httpd">/dev/null
```

```
          if [ $? == 0 ]
          then
           RC=${OPSTATE_ONLINE}
          else
           RC=${OPSTATE_OFFLINE}
          fi
          ;;
esac
exit $RC
```

Make sure to make the script accessible on all nodes with the same directory path.

RSCT resource definitions are created with the command **mkrsrc**. All resource characteristics can be provided in command line parameters, but the mkrsrc command also accepts a definition file in plain text. We will use the second approach with a definition file named apache1.def, which may look like the following:

```
PersistentResourceAttributes::
   Name="apache1"
   StartCommand="/cluster/scripts/apache start"
   StopCommand="/cluster/scripts/apache stop"
   MonitorCommand="/cluster/scripts/apache status"
   MonitorCommandPeriod=5
   MonitorCommandTimeout=5
   NodeNameList={"node01","node02","node03"}
   StartCommandTimeout=10
   StopCommandTimeout=10
   UserName="root"
   ResourceType=1
```

The resource definition can now be created with the **mkrsrc** command using the definition file.

```
mkrsrc -f apache.def IBM.Application
```

## Creating IP address resource apache1IP

The web server's IP address apache1IP is a separate IP address in the cluster and does not match any IP address assigned to the network adapters on each cluster node, that are made in system definitions outside of IBM Tivoli System Automation. The address for apache1IP is in contrast created by IBM Tivoli System Automation and is an additional alias address on an appropriate network adapter on the node where the web server resides. When the web server moves to a new location, the alias address is removed from the former node and recreated on the new node, where the web server is about to be restarted.

In this example apache1IP has the following attributes:

- IP 9.152.172.11
- Netmask 255.255.255.0
- The IP address may be created on any node in the cluster.

This time, we use command line parameters to the mkrsrc command to create the apache1IP resource:

```
mkrsrc IBM.ServiceIP \
  NodeNameList="{'node01','node02','node03'}" \
  Name="apache1IP" \
  NetMask=255.255.255.0 \
  IPAddress=9.152.172.11
```

Note that the command shown is split onto separate lines for readability only. In fact, the apache1IP has more attributes than the ones specified in the command

shown. We leave the rest of them to their default values, such as the ResourceType attribute, which marks the resource as “floating” by default.

Also note that the managed resources are not started/stopped by a third party like, for example, the Linux run level or manually by the operator.

## Step 3: Defining the automation policy

Before defining the automation policy there are some important things to get familiar with:

- By adding a resource into a resource group, this resource is automated and controlled by IBM Tivoli System Automation. This means that IBM Tivoli System Automation will manage to change the operational state (OpState) of the resource to the defined desired/nominal state (NominalState) of the resource group. If any resource automated by IBM Tivoli System Automation is manually started or stopped by an operator, IBM Tivoli System Automation will recognize this by monitoring this resource, and finally start or stop the resource again to bring it to the state defined in the automation policy.

  If a resource is defined in the automation policy, then IBM Tivoli System Automation will control this resource, and every manually operated action against this resource will usually trigger a reverse action of IBM Tivoli System Automation.
  This is also true for all resources on nodes that are excluded from automation (see the samctrl command described in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*).

  If at a certain point in time IBM Tivoli System Automation should not control the resources, then automation can be manually set with the following command:

  ```
  # samctrl -M T
  ```

  To return the control over the resources back to IBM Tivoli System Automation it has to be set in automatic mode again, using the following command:

  ```
  # samctrl -M F
  ```

  See the samctrl and lssamctrl commands described in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference* for further information.
- Resources (applications) that are controlled by IBM Tivoli System Automation can only be safely started and stopped by using the appropriate IBM Tivoli System Automation commands (chrg, rgreq or rgmbrreq). Only these commands will reflect the change of the desired state of a resource to IBM Tivoli System Automation. If a resource is started or stopped by its 'native' command, then IBM Tivoli System Automation still has the 'old' desired state of this resource in the automation policy, and finally will trigger a start or stop command against this resource to bring it again into the desired state.
- Nodes running a IBM Tivoli System Automation domain must not be maintained as without running IBM Tivoli System Automation. This is especially important for the commands halt and reboot. Before a node is rebooted or halted, it should be added to the automation exclude list, which will trigger IBM Tivoli System Automation to stop all resources running on that node and failing over these resources to another node. After all resources are Offline on that node, the node can be safely halted or rebooted . Failure to do so will result in an immediate reboot, if any 'critical resource' is running on the halted/rebooted node (see Chapter 9, “Protecting your resources – quorum support,” on page 137. The following command adds a node to the exclude list:

  ```
  # samctrl -u a <node-name>
  ```

  Now the node can be safely halted or rebooted. After the node is Online again in the domain, it has to be deleted from the exclude list with the following command:

  ```
  # samctrl -u d <node-name>
  ```

- IBM Tivoli System Automation will not start resources on nodes that are on the exclude list, even if the other nodes are Offline, so a node should only stay on the exclude list for the time of the maintenance. The exclude list can be checked using the following command:

  ```
  # lssamctrl
  ```

- IBM Tivoli System Automation domains with two nodes (or an even number of nodes) require a tie breaker to be setup. Without a tie breaker the second node will not take over resources that were running on a node that is crashed or rebooted. See Chapter 9, "Protecting your resources – quorum support," on page 137 for tie breaker types and how to setup these.

The following examples show, how the resources **apache1** and **apache1IP** are turned into IBM Tivoli System Automation managed resources (see "What is a managed resource?" on page 26) providing high availability for a web server in a cluster environment. See "Step 2: Defining RSCT resources" on page 16 how the resources apache1 and apache1IP were defined.

To turn resources apache1 and apache1IP into managed resources, they have to be added to a resource group. When this has been done, IBM Tivoli System Automation starts controlling the resource group and its included resources.

In most cases, it is not enough to automate a managed resource on its own, because the resources are often related to each other. For instance, both resources apache1 and apache1IP from our example must be made available on the same node. Such dependencies between managed resources must be described and defined with managed relationships (see "What is a managed relationship?" on page 53).

At last, an automation goal has to be provided, that is: should a managed resource be available/started in the cluster or should it be offline.

## Command overview

The following list provides an overview of the commands you use for defining policies. For a detailed description of the commands, refer to the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

**mkequ**
: Makes an equivalency resource

**chequ**
: Changes a resource equivalency

**lsequ**
: Lists equivalencies and their attributes

**rmequ**
: Removes one or more resource equivalencies

**mkrg**
: Makes a resource group

**chrg**
: Changes persistent attribute values of a resource group (including starting and stopping a resource group)

**lsrg**
: Lists persistent attribute values of a resource group or its resource group members

**rmrg**
: Removes a resource group

**mkrel**
: Makes a managed relationship between resources

**chrel**
: Changes one or more managed relationships between resources

**lsrel**
: Lists managed relationships

**rmrel** Removes a managed relationship between resources

**samctrl**
Sets the IBM Tivoli System Automation control parameters

**lssamctrl**
Lists IBM Tivoli System Automation controls

**addrgmbr**
Adds one or more resources to a resource group

**chrgmbr**
Changes the persistent attribute value(s) of a managed resource in a resource group

**rmrgmbr**
Removes one or more resources from the resource group

**lsrgreq**
Lists outstanding requests applied against resource groups or managed resources

**rgmbrreq**
Requests a managed resource to be started or stopped, or cancels the request

**rgreq** Requests a resource group to be started, stopped or moved, or cancels the request

**lssam** Lists the defined resource groups and their members in tree format

## Creating an equivalency for the network adapters

When a node in the cluster has multiple network attachments, not all of them might be equally suited to host the apache1IP address as an alias. An equivalency definition will specify the network adapters that can be used to carry the apache1IP address. Equivalency means that each of the adapters in the equivalency can provide the same required function regardless of its own unique characteristics. Since the web server should be startable on each node in the cluster, at least one of the adapters on each node has to appear in the equivalency.

An equivalency groups together a set of resources from another class. Network adapters belong to a class named IBM.NetworkInterface. There is no need to provide resource definitions for all the network adapters on the cluster nodes, since RSCT has a harvesting function which automatically creates appropriate resource definitions for many system defined resources.

The following command creates a so-called static equivalency named **netequ**, which contains a network adapter from each node of the cluster:

```
mkequ netequ IBM.NetworkInterface:eth0:node01,eth0:node02,eth0:node03
```

In a static equivalency, such as the one created with the command above, the members of the equivalency are specified at the creation time of the equivalency. The advantage of this equivalency type is that an order can be specified for the resources contained in the equivalency. A disadvantage is that network interfaces that are deleted from the cluster definition are also removed from the equivalency and will not be added automatically again if they are harvested later, thus resulting in a reduced member set.

A network interface is deleted from the cluster definition if its IP address is removed, or if the network interface is detached. If the network interface is

activated again later, it will be harvested by System Automation and again included in the cluster definition, but it will not be automatically added to the equivalency definition again.

To overcome this limitation, you can create an equivalency with a dynamic selection string using the following command:

```
mkequ -D 'Name like "eth0" ' netequ IBM.NetworkInterface
```

This definition includes all 'eth0' network interfaces available in the cluster. The selection string is evaluated each time System Automation has to find out on which node to start resources. The list of available 'eth0' resources may change as System Automation harvests new network interfaces in the cluster periodically, or when network interfaces that are undefined (IP address is removed or detached) are deleted from the cluster definition. However, equivalencies with a dynamic selection string cannot be ordered.

## Creating a resource group

A resource group is created with the **mkrg** command. The following command creates a resource group named **apacherg**:

```
mkrg apacherg
```

Both resources apache1 and apache1IP will be added to the resource group apacherg. This is done with the **addrgmbr** command. Adding the resources to the resource group turns them into managed resources:

```
addrgmbr -g apacherg IBM.Application:apache1
addrgmbr -g apacherg IBM.ServiceIP:apache1IP
```

## Defining relationships

There are two conditions that relate resources apache1 and apache1IP to one another. First, both resources must be started/available on the same node in the cluster. This is called a collocated relationship (see "Collocated relationship" on page 75). Furthermore, it is of no use to start the web server apache1 on a node, on which the IP address apache1IP has not been established yet. That is: apache1IP must be available before apache1 can be started.

IBM Tivoli System Automation provides a relationship type called DependsOn (see "DependsOn relationship" on page 64) that gathers both required conditions. A managed relationship is defined with the **mkrel** command. The following command creates a managed relationship named **apache1_dependson_ip1** that establishes the dependency of resource apache1 on the IP address apache1IP:

```
mkrel -p DependsOn -S IBM.Application:apache1 -G IBM.ServiceIP:apache1IP apache1_dependson_ip1
```

Our example needs a second relationship. In "Step 2: Defining RSCT resources" on page 16 we have created an equivalency of those network adapters, that can be used for aliasing the apache1IP address. This will be described in a second relationship called **apache1IP_dependson_netequ**, that ties apache1IP and the equivalency netequ together:

```
mkrel -p DependsOn -S IBM.ServiceIP:apache1IP -G IBM.Equivalency:netequ apache1IP_dependson_netequ
```

## Bringing a resource group online

When resources are added to resource groups, they become managed resources with a default automation goal of offline. This can be changed at the level of the resource group with the **chrg** command. To bring resource group apacherg online use the command:

```
chrg -o online apacherg
```

After your clusters and nodes have been created and configured, you can begin to use IBM Tivoli System Automation commands to:

- make, remove, change, and list *resource groups*. See Chapter 4, "Using resource groups," on page 31 for details.
- make, remove, and change the resource group *member resources*. See Chapter 3, "Using resources," on page 25 for details.
- make, remove, and change *equivalency resources*. See Chapter 5, "Using equivalencies," on page 47 for details.
- make, remove, change, and list *managed relationship* resources. See Chapter 6, "Using managed relationships," on page 53 for details.

A *complete list* of IBM Tivoli System Automation commands is provided in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

# Chapter 3. Using resources

This chapter describes how you use *resources*, in these main sections:
- "What is a resource?"
- "What is a managed resource?" on page 26
- "Attributes used by resources" on page 26

These are the IBM Tivoli System Automation commands that you use together with managed resources:

*Table 1. IBM Tivoli System Automation commands used with managed resources*

| Command | Description | For details, see the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*. |
|---|---|---|
| addrgmbr | Add one or more resources to a resource group | |
| chrgmbr | Change persistent attribute value of a managed resource in a resource group | |
| lsrg | List an already-defined resource group or its member resources | |
| rmrgmbr | Remove one or more resources from the resource group | |

## What is a resource?

A *resource* is any piece of hardware or software that has been defined to IBM's RMC (*Resource Monitoring and Control*) using either the:
- RMC **mkrsrc** ("Make Resource") command.
- "harvesting" function of RMC, in which resources are *automatically* detected and prepared for use with IBM Tivoli System Automation.

As described in "Components of IBM Tivoli System Automation" on page 6, IBM Tivoli System Automation uses as its basis the functionality of RMC. A resource is therefore sometimes referred to as an *RMC resource*.

Resources (adapter, program, disk, and so on) are controlled by a Resource Manager (abbreviated to **RM**).

## What is a resource class?

A resource class is a set of resources of the same type. For example, while a resource might be a particular file system or particular host machine, a resource class would be the set of file systems, or the set of host machines. A resource class defines the common characteristics that instances of the resource class can have (for example, all file systems will have identifying characteristics (such as a name), as well as changing characteristics (such as whether or not it is mounted). Each individual resource instance of the resource class will then define what its particular characteristic values are (for example, this file system is named "/var", and it is currently a mounted file system).

## What are resource attributes?

A resource attribute describes some characteristics of a resource. If the resource represents a host machine, its attributes would identify such information as the host name, size of its physical memory, machine type, and so on.

### What is the difference between persistent attributes and dynamic attributes?

There are two types of resource attributes – persistent attributes and dynamic attributes. The attributes of a host machine just mentioned (host name, size of physical memory, and machine type) are examples of persistent attributes – they describe enduring characteristics of a resource. While you could change the host name or increase the size of its physical memory, these characteristics are, in general, stable and unchanging.
Dynamic attributes, on the other hand, represent changing characteristics of the resource. Dynamic attributes of a host resource, for example, would identify such things as the average number of processes that are waiting in the run queue, processor idle time, the number of users currently logged on, and so on.

## What is a managed resource?

A resource becomes an *IBM Tivoli System Automation managed resource* (referred to simply as a *managed resource*) as soon as the resource has been inserted in an IBM Tivoli System Automation resource group. This is done using the IBM Tivoli System Automation **addrgmbr** command (described in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*). From this point onwards, the resource can be managed using the IBM Tivoli System Automation commands and programs.

Managed resources are supplied in IBM Tivoli System Automation resource class IBM.ManagedResource.

## Working with resources

Note that IBM Tivoli System Automation does not allow you to start or stop **resources** directly.

Resource starting and stopping is based on the setting of the NominalState attribute of a **resource group**. For example, setting the NominalState attribute of a resource group to ″Online″ indicates that you wish to start the resources in the resource group. Setting the NominalState attribute of a resource group to ″Offline″ indicates that you wish to stop the resources in the resource group. See the description in “Starting and stopping a resource group” on page 44 and “NominalState attribute” on page 37.

## Attributes used by resources

Resources can have the following attributes:

**NodeNameList**
: indicates on which nodes the resource is allowed to run. This is an attribute of an RSCT resource.

**SelectFromPolicy**
: defines the list of nodes the resource is available on. This is an attribute of a managed resource.

**ResourceType**
: indicates whether the resource is allowed to run on multiple nodes, or a single node. This is an attribute of an RSCT resource.

**OpState**
: specifies the operational state of a resource or a resource group. This is an attribute of an RSCT resource.

## NodeNameList attribute

The NodeNameList persistent attribute represents the set of nodes on which the resource can run.

The nodes resources started on by IBM Tivoli System Automation are controlled by the order of the nodes in the NodeNameList of the resources. By default, each floating resource will be placed on the first node in its node list where it can be started with respect of all its relationships to other resources. This behavior can be modified by using the SelectFromPolicy attribute described below. Floating resources which should be collocated are placed before independent floating resources (these are resources which do not have relationships to other resources) and anticollocated floating resources.

If there are collocated resources with different node lists, the resources are placed on the node which is chosen by the majority of resources. In a tie situation, one node is chosen randomly.

## SelectFromPolicy attribute

As described above, the NodeNameList persistent attribute defines the list of nodes the resource is available on. This list is ordered, either in the sequence as specified by the user or in the sequence the resource manager owning the resource put them in. The SelectFromPolicy attribute gives the user more flexibility. It allows to tell IBM Tivoli System Automation which algorithm to use when selecting a node from the list. This can be either ordered, meaning that IBM Tivoli System Automation always starts from the beginning of the list when determining the next available node, or any, meaning that IBM Tivoli System Automation picks a node randomly.

## ResourceType attribute

The ResourceType attribute is either defined by the resource manager or during creation of the resource. The ResourceType persistent attribute specifies whether a resource is:

- Serial fixed (its NodeNameList attribute contains a *single* node entry).

  **Fixed resource**

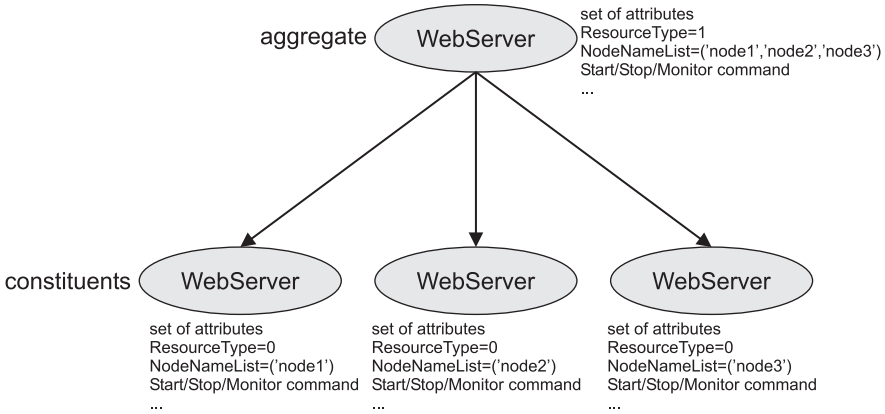
  A fixed resource is a resource of which there is only a single instance within the cluster. It is defined upon a single node, and that is where it runs. It represents one entity such as a process, a mount point, or a network adapter.

  If you want to define an application which runs on more than one node in the cluster at a time, you must define one fixed resource for each node, specifying the same attributes (for example, name, start/stop/monitor commands) but a different node name for each fixed resource.
- Serial floating (its NodeNameList attribute contains *one or multiple* entries). Although multiple nodes are defined for possible use, only one instance of the resource may be active at any time. For example, an IP address that can be moved from one machine to another is a floating resource; although multiple machines may have use of the IP address at some point, only one machine at a time will use it.

  **Floating resource**

  A floating resource is a resource which can run on several nodes in the cluster. A floating resource is represented in RMC in the following way:

You have one aggregate resource and one constituent resource on each node belonging to the aggregate resource.




The aggregate resource has a ResourceType attribute value of 1. The set of nodes where the resource should be able to run is defined in the NodeNameList attribute of the aggregate resource. The other attributes of this resource are defined by the resource manager and its class definition.
If you create a floating resource you create the aggregate resource. The resource manager responsible for this type of resource will create constituent resources on each node the resource is supposed to run on. The constituent resources have their own values of the attributes. The ResourceType of a constituent resource is 0 (a fixed resource), and the NodeNameList contains one node only. At time of creation the other attributes have identical values as the aggregate resource.

The following happens if you change attributes of a floating resource:

- A change of the NodeNameList of the aggregate resource causes deletion or creation of constituent resources.
- If you change an attribute of the aggregate resource this changes the according attributes of all constituent resources.
- If you change an attribute of a constituent resource this affects the constituent resource only and is not conveyed to other constituent resources or the aggregate resource.

A floating resource represents an automatable entity such as an application or a service ip address which can run on several nodes.

**Note:** A floating resource is labelled as move group in the operations console.

## OpState attribute

RMC uses the OpState dynamic attribute to specify the operational state of a resource. It is mandatory for resources added to a resource group.

These are the possible values that the OpState attribute can have:

**Offline** The resource is not started.

**Pending Online**
The resource has been started, but is not yet ready for work

**Online** The resource is ready for work.

**Pending Offline**
The resource is in the process of being stopped

Some of the operational states indicate problems:

**Failed Offline** The resource is broken and cannot be used. You have to reset the resource when you have fixed it.

**Stuck Online** The resource was being started, but did not become ready for work within the expected time interval and cannot be brought offline. Another possibility is that the resource was online, but an offline request could not bring it offline.

**Unknown** IBM Tivoli System Automation is unable to obtain reliable state information from the RMC managing the resource.

**Note:** You might have to reset a resource that has the Failed Offline state. To do so, use the RMC command **resetrsrc**. For details, refer to the man page for this command.

When a node of a resource is Offline, the resource is considered to be Failed Offline, even though its operation state at that point is Unknown. IBM Tivoli System Automation can do this because it has separate state data for the resources node.

## Nominal state of a resource

Resources do not have nominal state information. You cannot set the nominal state of a resource directly. Resources must be defined within the resource groups. Each resource group has a nominal state. This is either Online or Offline and tells IBM Tivoli System Automation whether the resources within the resource group should be Online or Offline at this point in time. You can change the resource group nominal state value.

## Attributes used by resources

# Chapter 4. Using resource groups

This chapter describes how you use resource groups, in these main sections:
- "What is a resource group?"
- "Attributes used by resource groups" on page 34
- "Making (creating) a resource group" on page 42
- "Changing attributes of a resource group" on page 44
- "Removing a resource group" on page 45
- "Listing a resource group or its resource members" on page 43
- "Adding a member resource to a resource group" on page 42
- "Removing a member resource from a resource group" on page 44
- "Changing the attributes of resource group members" on page 44

**Related Section:**

- "Events that might allow a resource group to become Online" on page 90

Table 2 lists the IBM Tivoli System Automation commands that you use together with resource groups. For details, see the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

*Table 2. IBM Tivoli System Automation commands used with resource groups*

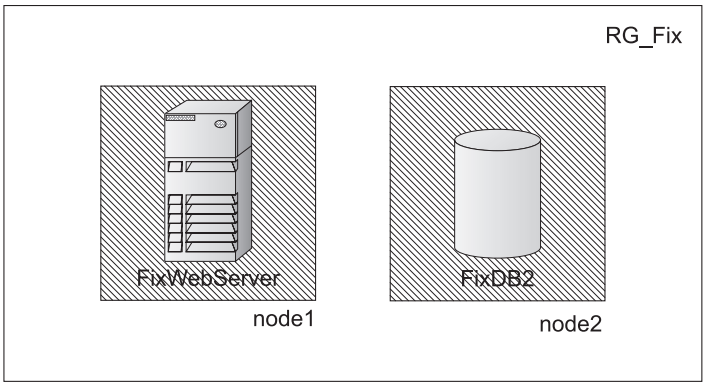
| Command | Description |
|---|---|
| mkrg | Make a resource group |
| rmrg | Remove a resource group |
| chrg | Change persistent attributes of a resource group (including starting and stopping a resource group) |
| lsrg | List one or more resource groups |
| addrgmbr | Add member resources to a resource group |
| rmrgmbr | Remove member resources from a resource group |
| chrgmbr | Change attributes of the member resources of the resource group |
| rgreq | Start, stop, cancel, or move a resource group |
| rmrgmbr | Remove one or more resources from a resource group |

## What is a resource group?

The central unit in IBM Tivoli System Automation are the resource groups. They are a logical container for a collection of resources that can be treated as one logical instance. One aspect of resource groups is that you can use them to control all of their members collectively. For example, if you set a resource group's NominalState to Online all members are started and kept online. Vice versa, if you set the NominalState to Offline all members are stopped and kept offline. Another aspect of resource groups is that it is possible to monitor their OpState which provides a consolidation of the OpStates of the individual resource group members. Members of a resource group can be of type:

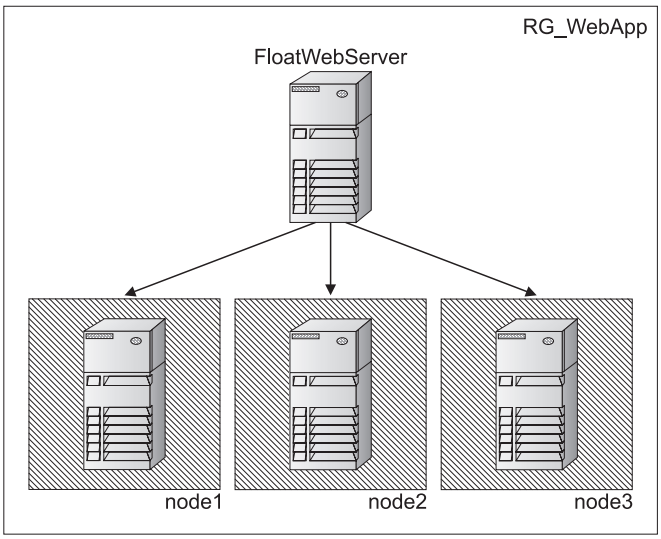
- Serial fixed.
- Serial floating.
- And even resource groups itself which means that nested groups can be defined.

An example for a resource group containing fixed resources is a resource group RG_Fix which contains serial fixed resources. These are a web server FixWebServer which can only run on node1, and a database resource *FixDB2* located on node2.




In order to start both resources FixWebServer and FixDB2, set the NominalState of RG_Fix to Online. This example also shows that IBM Tivoli System Automation can handle resource group members of a resource group which are distributed on different nodes in a cluster.

An example for floating resource group members is the following: A web server apache1 could run either on node1, node2, or node3. The resource group RG_WebApp would look very similar except that the web server could be started on either of the three nodes




This example shows that resource groups can contain a mixture of members of different resource types.

The concept of resource groups is very powerful as it allows defining resource groups as members of other resource groups. An example is resource group RG_A which has as members resource A, which is a fixed resource, and RG_WebApp, the resource group from the previous example. Nested resource groups allow structuring complex environments in several layers. The nesting level is 50.

Another flexibility of the resource groups functionality is that all kinds of relationships like start /stop relationships and location constraint relationships can

be defined with resource groups as source or target resource. Furthermore it is allowed that resource group members can be part of such relationships as source or target resource.

Resource groups are defined in IBM Tivoli System Automation resource class IBM.ResourceGroup.

## Rules for using resource groups

These are the rules for using resource groups:

1. A resource group *cannot* contain an equivalency or vice versa.
2. A resource can only be in one group.
3. A member cannot be in a group and in an equivalency.
4. The nesting level of a resource group is limited to 50.
5. The number of resources linked by groups or relationships is limited to 100.

## Attributes used by resource groups

A resource group provides the following persistent RSCT attributes which can be defined by the user:

**AllowedNode** limits the nodes on which the resource group members are allowed to be started.

**MemberLocation**
defines if all members of a resource group have to be collocated or not. Collocated means that all members have to run on the same node. None means that the members are not dependent from each other and can arbitrarily run on the nodes on which they are defined.

**Name** defines a unique name for a resource group.

**NominalState** is the desired state of the resource group. IBM Tivoli System Automation tries to bring up and keep the resource group in this state.

**Priority** defines the importance of a resource group in a conflict situation.

**ExcludedList** defines a list of nodes which are temporarily excluded from the node list of the group.

**ActivePeerDomain**
is the name of the peer domain the group is defined to.

**Description** can contain descriptive text about the resource group.

**InfoLink** can be used to specify a URL of an HTML page where the operator can find additional information about the resource.

**Owner** provides information about the owner of the resource group, for example, the name and telephone number of a responsible person.

**Subscription** can contain subscription information from end-to-end management.

**Note:** The persistent attributes described in this section can only be modified if the resource group containing them is **Offline**. An exception are the NominalState attribute and the informational attributes Description, InfoLink, and Owner, which can also be modified when the resource group is **Online**.

A resource group provides the following dynamic attributes:

**OpState** Specifies the aggregate operational state of the collection of managed resources.

**TopGroup** Shows the name of the top level resource group of a resource group.

**AutomationDetails**
Shows IBM Tivoli System Automation internal states of the resource group.

**MoveStatus** Shows the progress of a move of a resource group initiated by a rgreq command.

**ConfigValidity**
Shows if a policy has become invalid.

## AllowedNode attribute

You use the AllowedNode parameter to define a set of nodes in a cluster on which the members of a resource group are limited to run.

You can choose between the following parameters:

**All** is the default value. It means that no limitation is made by the resource group. It can run on all nodes in the cluster.

**One node** defines a specific node on which all resource group members have to run on. If the specified node is removed from the cluster at a later time, then AllowedNode will default to All.

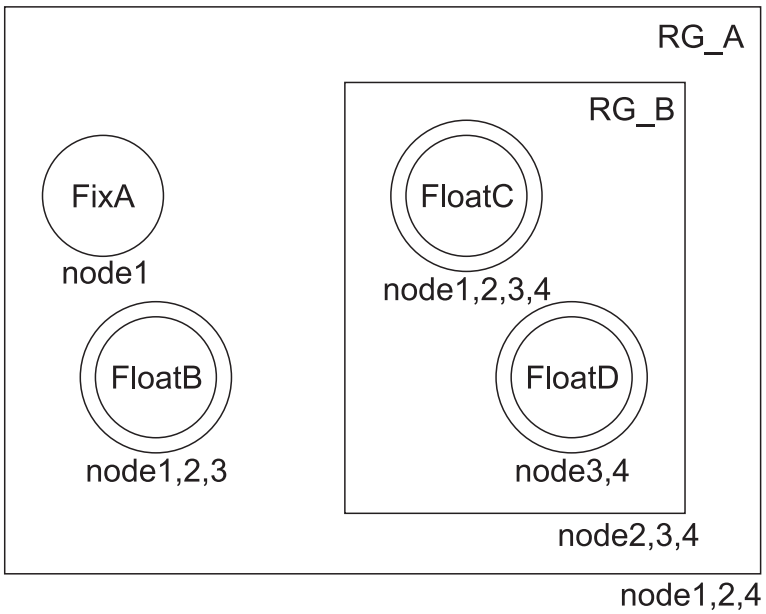
**Equivalency of nodes**
contains a set of nodes where the resource group members are limited to run on. Only static equivalencies are allowed. See also Chapter 5, “Using equivalencies,” on page 47.

### Node limitation aspect of AllowedNode parameter

In specific cases it might be necessary to limit a member of a resource group to run on a set of nodes. For example, when a floating resource is defined a NodeNameList has to be specified which is in general independent from the IBM Tivoli System Automation usage. The NodeNameList of the floating resource is used by the resource managers (for instance GblResRM) which own a floating resource. For them the NodeNameList defines on which nodes the constituents of a floating resource could potentially run. Whereas the AllowedNode attribute belongs to a resource group parameter that has different resource group members which are floating resources. Here the AllowedNode attribute allows to limit the nodes on which the resource group members are allowed to run. This means for resource group members of type:

- **Fixed resource**, that the NodeNameList which contains only one node on which the fixed resource is located has to be part of the AllowedNode parameter.
- **Floating resource**, that the intersection of the NodeNameList and the AllowedNode parameter defines the set of nodes on which the floating resource is started and controlled by IBM Tivoli System Automation.
- **Resource Group**, that an intersection of the AllowedNode parameter of the inner and the outer group derives a resulting list of allowed nodes for the inner group. The resulting list defines the nodes on which the members of the inner resource group are allowed to run on.

The following example explains the behavior of the AllowedNode parameter on the node limitation: Given an outer resource group RG_A with a member FixA with a NodeNameList = {node1}, a floating member FloatB with a NodeNameList ={node1, node2, node3} and a resource group RG_B with AllowedNode = {node2, node3, node4}. The AllowedNode list of RG_A defines {node1, node2, node 4}. RG_B contains FloatC with a NodeNameList of {node1, node2, node3, node4} and FloatD with NodeNameList of {node3, node4}.




The result of this scenario is that:

- FixA can only be started by IBM Tivoli System Automation on node1.
- FloatB can only be started by IBM Tivoli System Automation on node1 and node2.
- RG_B's members are limited to run on node2 and node4.
- FloatC can only be started by IBM Tivoli System Automation on node node2 and node4.
- FloatD can only be started by IBM Tivoli System Automation on node4.

The AllowedNode list of an inner group is not affected when its outer group has no node limitation due to its intersection of the own AllowedNode parameter and the intersection with the AllowedNode list of its outer group

## MemberLocation attribute

You use the MemberLocation persistent attribute to specify the default location between resources in the resource group. Valid values are either:

**Collocated (the default)**
Collocated means that all members have to run on the same node. If you specify Collocated, you cannot apply the Affinity, AntiAffinity, and AntiCollocated managed relationships between the resources in the resource group to specify on which nodes the member resources should be located. See Chapter 6, "Using managed relationships," on page 53 for a detailed explanation of relationships.

**None** None means that the resource group implies no restriction regarding the location of its members.

The MemberLocation attribute is applied to *all* member resources of the resource group.

If resource groups are nested, the value of the MemberLocation attribute of the outer resource groups must allow the inner-group(s) specification. Therefore, if a resource group's MemberLocation attribute is collocated, only collocated resource groups are allowed as members.

## Name attribute

The Name of a resource group has to be unique in a cluster.

## NominalState attribute

You use the NominalState persistent attribute to control a resource group. By setting the NominalState of a resource group to Online all of its members are started and kept online.

The NominalState attribute can be either Online or Offline. A NominalState of Offline causes that all resource group members are stopped and kept offline. There are exceptions:

1. If resource groups are nested, a NominalState of online of an outer group overrules a NominalState of offline of an inner group and starts these inner groups. You cannot stop such an inner group separately.
2. If resource groups are nested, a NominalState of offline of an outer group will not overrule a NominalState of online of an inner group. A NominalState of online of an inner group will start this group and groups contained in this inner group. The outer groups' OpState will change, but the other group members remain untouched.
3. Cross group dependencies might force individual group members to be started. See "Relationships for start / stop behavior" on page 56.

The default value for the NominalState of a resource group is Offline.

This attribute can be modified at any time.

**Note:** You can modify this attribute with the **chrg –o** command (see the description of the chrgcommand in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*). The numeric value of this attribute is

**1 Offline**

**0 Online**

## Priority attribute

You use the Priority persistent attribute to specify the relative priority of this resource group in relationship to other resource groups.

The Priority attribute is used to resolve conflicts when resource groups are being started that have conflicting *managed relationships* (described in Chapter 6, "Using managed relationships," on page 53) to other started or online resources. These conflicts may be between the resource group or a managed resource contained in that group and any other started or running resource.

For instance, you have a cluster with only one node online, and two resource groups with an AntiCollocated relationship against each other. This means that the resource groups must never be started on the same node at a time. IBM Tivoli System Automation now uses the value of the Priority attribute of the resource groups to find out which of the resource groups should be online, and which cannot be online because of the conflicting AntiCollocated relationship.

If a lower priority active resource group prevents the activation of a higher priority resource group because of conflicting relationships, the lower priority resource group is stopped in order to allow the activation of the resource group with higher priority to proceed.

If resource groups are nested, the outer resource group must have a higher or equal priority than any of the inner resource groups.

The default value of the Priority attribute is zero, which is the lowest value. The maximum value is 200.

The actual priority of a resource group is computed from the value of the priority attribute and an adjustment value which is determined by the current observed state of the group (see "AutomationDetails attribute" on page 40). The adjustment value for groups that are online is +10, to make sacrificing started resources harder, and -10 for groups that are failed offline. The actual priority value can assume values from -10 to +210. (For more information on the meaning of the priority, see "Location relationship resolution: Binding algorithm" on page 87.)

**Hint**: IBM Tivoli System Automation also uses the Mandatory attribute of the managed resources (described in "Attributes used for resource group members" on page 41) to determine which resources will be started in a conflict situation. In case of nested resource groups, the resource groups that are non-mandatory members should have a lower priority than mandatory members. Otherwise Mandatory members may be discarded.

## ExcludedList attribute

You use the ExcludedList attribute to temporarily exclude one or a list of nodes from the node list of the group. When excluding a node, the resources residing on the node being excluded are not automatically forced offline. The move must be triggered by means of the rgreq command.

This means that placing a node in the exclude list causes IBM Tivoli System Automation to not consider the node as a potential candidate for hosting the resource. It can be used to gradually and non-disruptively move resources away from a node in preparation for an EXCLUDE (via the samctrl command) at a later point in time.

The following rules apply:

1. Excluding a node means for a fixed resource group member that the resource cannot be started anymore.
2. Excluding a node means for a floating resource group member that its constituent on that node cannot be started any more.

## ActivePeerDomain

This attribute shows the name of the RSCT Peer Domain the group is defined in.

## Description

This attribute can contain descriptive text about the resource group. This attribute only serves informational purposes and does not affect automation functions.

## InfoLink

This attribute can be used to specify a URL of an HTML page where the operator can find additional information about the resource. This attribute only serves informational purposes and does not affect automation functions.

## Owner

This attribute provides information about the owner of the resource group, for example, the name and telephone number of a responsible person. This attribute only serves informational purposes and does not affect automation functions.

## Subscription

This attribute contains subscription information from end-to-end management.

## OpState attribute

IBM Tivoli System Automation uses the OpState dynamic attribute to specify the aggregate operational state of the collection (of managed resources). It is determined from the individual operational states of the member resources of the resource group. These are the possible values that the OpState attribute can have:

| Status | Value | Description |
|---|---|---|
| **Unknown** | 0 | |
| **Online** | 1 | Specifies that all of the Mandatory member resources are Online. The Non-Mandatory member resources are ignored. |
| **Offline** | 2 | Specifies that all of the member resources are Offline. |
| **Failed Offline** | 3 | Specifies that one or more member resources contained in the resource group, are FailedOffline. In this case, all resources contained in the resource group will be set to Offline. |
| **Stuck Online** | 4 | Specifies that a member resource is stuck online. |
| **Pending Online** | 5 | Specifies that a Start command is executed (the resource group's NominalState attribute is set to Online). The resource group must begin processing an Online action. |
| **Pending Offline** | 6 | Specifies that an Offline action has been initiated. |

## TopGroup attribute

You use the TopGroup dynamic attribute to view the top level resource group of a resource group.

In IBM Tivoli System Automation resource groups can be members of another resource group, meaning that resource groups can be nested. The TopGroup attribute shows the name of the top level resource group of the current group. The attribute can be displayed using the **lsrg** command as described in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference* and shown below.

**Note:** When using the lsrg –g command to query a resource group, the NominalState of the top group is shown in the output for user convenience together with the TopGroup attribute. The output looks similar to:

```
TopGroup             = apacherg
TopGroupNominalState = Offline
```

## AutomationDetails attribute

This attribute shows System Automation internal states of the group. These states include:

**CompoundState**
Overall status of the resource group including group dependencies. An example is ″Satisfactory″ – resource/group has reached the requested user status.

**DesiredState** User requested status of the resource group. An example is ″online″ – user requested that the resource group should be online.

**ObservedState**
Real status of the resource group from an automation point of view. An example is ″online″ – the resource group is currently online.

**BindingState** Status indicating if the resource group is bound to a specific system. An example is ″bound″ – the resource group is currently bound to a specific system.

**AutomationState**
Status indicating if the resource group is currently being automated. An example is ″Idle″ – IBM Tivoli System Automation is currently not trying to start or stop the resource group.

**ControlState** Status indicating if the resource group can be controlled by automation. An example is ″startable″ - it is currently possible to start this resource group.

**HealthState** currently not used, reserved for future releases.

To show the automation details as indicated above you use the lsrg command with the –A d and the –V option. For example, to show the automation details of a resource group named ″apacherg″, you use the command:

```
lsrg –A d –V –g apacherg
```

## MoveStatus

Shows the progress of a move of a resource group initiated by a rgreq command.

To show the move status you use the lsrg command with the –A d and the –V option. For example, to show the move status of a resource group named ″apacherg″, you use the command:

```
lsrg –A d –V –g apacherg
```

## ConfigValidity

After a policy was established, several things can happen to make the policy invalid. This attribute shows the reason for invalidity.

For example, if a node is removed from the peer domain which was the only common node for the members of a collocated resource group, the resources can no longer be started. This will be indicated by the ConfigValidity attribute.

## Attributes used for resource group members

In addition for each resource group member a user has to define persistent attribute:

**Mandatory** Defined for each resource group member and specifies whether it is mandatory for the group. Alternatively a member can also be non-mandatory.

**MemberOf** The name of the resource group of which the resources are members.

**SelectFromPolicy**
Used to tell IBM Tivoli System Automation where floating resources should be preferably started.

**ConfigValidity**
This attribute is reserved for future use.

### Mandatory attribute

You use the Mandatory persistent attribute to specify whether a managed resource is Mandatory or Non-Mandatory.

When a resource group is started, all managed resources within that group that are Mandatory must also be started. Managed resources that are Non-Mandatory (whose Mandatory attribute is set to False) might not be started when a conflict exists. If a managed resource that is Mandatory fails, the entire resource group is stopped and started on another node, but if a non-mandatory member of a resource group fails, the resource group stays Online on that node.

Resources that are members of a resource group are implicitly mandatory unless this attribute value is explicitly set to False.

Member resources whose Mandatory managed resource attribute is False may be sacrificed in order to activate the resource group.

### MemberOf attribute

Indicates that the resource is contained in a resource group resource. The MemberOf persistent attribute is generated implicitly when resources (including nested resource groups) are added to a resource group. The MemberOf attribute is used to determine the set of resources to be started and stopped when the resource group is activated or deactivated (either explicitly through a stop order, or implicitly through a non-recoverable member resource failure). A resource can be a member of one and only one resource group.

# Defining and administering a resource group

## Making (creating) a resource group

To make (create) a resource group you use the **mkrg** command, in which you define to IBM Tivoli System Automation:

- Where the resource group is allowed to run.
- The relative importance of the resource group in relation to other resource groups (using the *Priority* attribute, as explained in "Priority attribute" on page 37).
- The *Location* relationship among the member resources of the resource group (explained in "Location relationships" on page 73).

Newly-created resource groups will default to an NominalState of Offline. This allows a user or administrator to fully configure the resource group and its resources.

For example, to define a new resource group called **apacherg2** with location relationship "None" and allowed node name "node03", you would enter:

```
mkrg -l None -n node03 apacherg2
```

For further details, see either the **mkrg** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

To establish the node list for a resource group, you can use either:

- **mkrg** command, to create a new resource group.
- **chrg** command, for an already-existing resource group.

To specify the node list to be used with the above two commands, you can either:

- Enter a node name when using the **mkrg**/**chrg** command.
- Establish as an *equivalency*, the set of nodes on which the resource group can be activated. This must be done before the node list is established. You then use the **mkrg**/**chrg** command, and the required equivalency will be attached to the resource group. (For details of equivalencies, see Chapter 5, "Using equivalencies," on page 47).

## Adding a member resource to a resource group

To add one or more new member resources to a resource group, you use the **addrgmbr** command.

**Notes:**

1. A member resource cannot be included in more than one resource group at the same time.
2. A member resource cannot be in a resource group *and* in an equivalency at the same time.

For example, to add member resource apache1, belonging to resource class IBM.Application, to a resource group apacherg2 , you would enter:

```
addrgmbr -g apacherg2 IBM.Application:apache1
```

For further details, see either the **addrgmbr** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Listing a resource group or its resource members

You can use the following commands to list resource groups and their members:

**lssam** Lists resource groups and their members in tree format. For detailed information about the command, see the **lssam** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

**lsrg** Lists resource groups or the members of a resource group. This section gives you an overview of how the command is used and provides some usage examples.

For further details, see either the **lsrg** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

To list a resource group or its members, you can use the **lsrg** command. If the resource group name is omitted, all of the resource groups are listed. If the resource group name is specified with the **-m** option, the member resource names and resource classes are listed.

If the *Attr* parameter is specified, then the attributes specified for the resource group are listed.

Three resource groups are defined in the following examples:

**Example 1:** If command **lsrg** is entered, this type of information is displayed:

```
Resource Group Names:
apacherg2
apacherg3
apacherg4
```

**Example 2:** To list all members of all resource groups, enter:

```
lsrg -m
```

This information is then displayed:

```
Displaying Member Resource information:
Class:Resource:Node[ManagedResource]  Mandatory MemberOf  OpState  WinSource  Location
IBM.Application:apache1               True      apacherg2 Offline  Nominal    node03
```

**Example 3:** apacherg2 contains one resource. To list the members of apacherg2, this command is entered:

```
lsrg -m -g apacherg2
```

This information is then displayed:

```
Member Resource 1:
        Class:Resource:Node[ManagedResource] = IBM.Application:apache1
        Mandatory                            = True
        MemberOf                             = apacherg2
        OpState                              = Offline
```

**Example 4:** To list the attributes of a resource group apacherg2, this command is entered:

```
lsrg -g apacherg2
```

This information is then displayed:

```
Resource Group 1:
        Name           = apacherg2
        MemberLocation = None
        Priority       = 0
        AllowedNode    = node03
        NominalState   = Offline
        OpState        = Offline
```

## Starting and stopping a resource group

To start or stop a resource group you set the NominalState attribute of the resource group to online or offline respectively. Use the **chrg** command to do this.

For example, to start a resource group called apacherg2, you would enter:

```
chrg -o online apacherg2
```

For further details, see either the **chrg** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Changing attributes of a resource group

To change the persistent attribute values of one or more resource groups, you use the **chrg** command. The name of a resource group can also be changed with this command, using the **-c** option.

**Example 1**: to change the location relationship of a group apacherg2 to collocated, you would enter:

```
chrg -l collocated apacherg2
```

**Example 2**: to change the name of a group apacherg3 to apacherg4, you would enter:

```
chrg -c apacherg4 apacherg3
```

For further details, see either the **chrg** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Changing the attributes of resource group members

To change the attributes of the specified member resources, you use the **chrgmbr** command.

This command also allows you to specify changes to the Mandatory attribute of a managed resource by using the **-m** option , and to change the resource group to which the resource belongs by using the **-c** option.

For example, to change the resource group to which member resource apache2 of resource class IBM.Application belongs, from the current resource group apacherg2 to resource group apacherg3, you would enter:

```
chrgmbr -c apacherg3 -g apacherg2 IBM.Application:apache2
```

For further details, see either the **chrgmbr** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Removing a member resource from a resource group

You use the **rmrgmbr** command to remove:

- all the member resources of a specified resource group.
- only the specified member resources of the specified resource group.

- the member resources that match a selection string.

IBM Tivoli System Automation also ensures that any associated managed relationship or equivalency is also updated.

For example, to remove member resource apache2 that belongs to resource class IBM.Application, from resource group apacherg3, you would enter:

```
rmrgmbr -g apacherg3 IBM.Application:apache2
```

For further details, see either the **rmrgmbr** man page, or the command description in the*IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Removing a resource group

To remove one or more resource groups, you use the **rmrg** command. The resource groups are specified either using the *Resource_group* parameter, or by matching a selection string. The member resources associated with the removed resource groups, are also removed by IBM Tivoli System Automation. If the resource group to be removed is still online, the resource group is not removed. Note that relationships where these deleted resource groups are the source, are also deleted.

This implies that all resource groups that are nested within the resource group to be removed are also removed recursively. If you want to prevent contained resource groups from being deleted recursively, proceed as follows:

1. Remove these resource groups as members from the resource group to be removed using the **rmrgmbr** command.
2. Remove the containing resource group.

For example, to remove resource groups called apacherg2 and apacherg3, you would enter:

```
rmrg apacherg2 apacherg3
```

For further details, see either the **rmrg** man page, or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

# Chapter 5. Using equivalencies

This chapter describes equivalencies in these main sections:
- "What is an equivalency?"
- "Attributes used by equivalencies" on page 49
- "Rules for using equivalencies" on page 48
- "Making (creating) an equivalency" on page 50
- "Changing an equivalency" on page 51
- "Removing an equivalency" on page 51
- "List one or more equivalencies" on page 50

Table 3 lists the IBM Tivoli System Automation commands that you use for equivalencies. For details, see the command descriptions in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

*Table 3. IBM Tivoli System Automation commands used with equivalencies*

| Command | Description |
|---|---|
| mkequ | Make an equivalency resource |
| rmequ | Remove an equivalency resource |
| chequ | Change an equivalency resource |
| lsequ | List equivalency resources |

## What is an equivalency?

An equivalency is a collection of resources that provide the same functionality. An equivalency consists of a set of fixed resources from the same resource class.

For example, network adapters might be defined as equivalencies. If one network adapter goes offline, another network adapter can take over the processing from the offline adapter.

Equivalencies are also used for establishing a resource group's node list. From this equivalency, one or more resources can be selected to satisfy a managed relationship. But only one member is started on a node to satisfy a managed relationship. For details of managed relationships, see Chapter 6, "Using managed relationships," on page 53.

There are two types of equivalency:

1. Equivalencies with a static membership list. An equivalency of this type contains a fixed set of resources which a user has explicitly added to the equivalency.
2. Equivalencies with a SelectString list. An equivalency if this type dynamically determines at run-time which resources are contained within the equivalency. If RMC resources are created which match the dynamic select string, these are automatically contained within the equivalency. Specifying a policy is not reasonable for this type of equivalency as the resources are not ordered.

Additionally, the following types of equivalency can be used to effect a particular automation behavior:

**Failback**
Can be used for ordered equivalencies only. Equivalencies of this type show the following behavior:

IBM Tivoli System Automation ensures that all resources of the equivalency run on the node hosting the first member of the equivalency. If this member goes Offline, IBM Tivoli System Automation starts the resources on the node hosting the second member of the equivalency. When the first member has come back Online, IBM Tivoli System Automation stops all dependent resources and restarts them again on the node hosting the first member of the equivalency.

Note that a move request against a resource group that has a relationship to an equivalency of type Failback does not result in the expected behavior: The move will take place, however, immediately after the move is complete, the resource group is failed back to the original node!

**NoFailure**
Equivalencies of this type show the following behavior:

Resources that have a dependency to the equivalency will not be set to a Failed Offline state if they do not reach the Online operational state within the timeout interval. This is useful for resources that take a long time to start. The meaning is similar as if the StartCommandTimeout of a resource were set to infinite, but differs in that these resources can be automated again after the Online timeout is reached, while resources with very high StartCommandTimeout values cannot be automated until the StartCommandTimeout is reached because they are in a Waiting state.

**Shadow Resources/Shadow equivalencies**
A shadow resource monitors the OpState of another resource. IBM Tivoli System Automation only evaluates the OpState of shadow resources but neither starts nor stops them. Shadow resources are used to define relationships from a floating resource to one of two or more fixed resources running on different nodes. For a detailed description, see "Using shadow resources" on page 178.

Equivalencies are supplied in IBM Tivoli System Automation resource class IBM.Equivalency.

## Rules for using equivalencies

These are the rules for using Equivalencies:

1. A resource can be a member of either an equivalency or a resource group, but not both.
2. A resource may be in more than one equivalency.
3. The specified resources must all be from the same resource class.
4. Equivalencies cannot be members of an equivalency.
5. Resource groups cannot be members of an equivalency.
6. Equivalencies cannot be members of a resource group.
7. An equivalency that satisfies a relationship for an active resource cannot be modified.
8. An equivalency can only be the *target* of a managed relationship (it cannot be the *source* of a managed relationship).
9. The members of an equivalency must be fixed resources. Floating resources are not allowed.

# Attributes used by equivalencies

## MemberClass attribute

IBM Tivoli System Automation uses the MemberClass persistent attribute to determine the resource class of all the member resources.

## Membership attribute

IBM Tivoli System Automation uses the Membership persistent attribute to determine the set of resources contained within the equivalency. If a Membership attribute is specified, no SelectString attribute is allowed.

## SelectString attribute

IBM Tivoli System Automation uses the SelectString persistent attribute to dynamically determine the resources contained within an equivalency. If resources matching the selection string are inserted in, or removed from, the system, then IBM Tivoli System Automation automatically modifies the equivalency. If a SelectString attribute is specified, no Membership attribute is allowed.

## SelectFromPolicy attribute

IBM Tivoli System Automation uses the SelectFromPolicy persistent attribute to determine the policy to be used in making a selection from the equivalency. You can modify this attribute when the resources referencing the equivalency are offline. Possible policies:

**Any (the default)**
In this case, if a resource contained within an equivalency fails, IBM Tivoli System Automation chooses any resource without referring to a pre-specified order of selection.

**Ordered** In this case, if a resource contained within an equivalency fails, IBM Tivoli System Automation always starts from the beginning of the selection list.

**Note:** An Ordered policy may not be used when a dynamic SelectString is used.

There are additional settings for the SelectFromPolicy persistent attribute, which can be combined with the Any or Ordered setting to effect a particular automation behavior:

**Failback**
The Failback option can only be specified for an Ordered policy. If the option is specified, IBM Tivoli System Automation will fail back the resources to the node hosting the first member of the equivalency when the first member becomes available again.

Note that this feature has an impact on move requests against resource groups (see also “Moving resource groups with the rgreq command” on page 176): If an equivalency with a Failback policy is involved in a move request, the move of the resource group is performed, but immediately after the resources are Online on the target node, the Failback option of the equivalency causes the resource group to be moved back to the original node.

**NoFailure**
If the NoFailure option is specified, resources that have a DependsOn

relationship to the equivalency will not be set to Failed Offline if an Online request against these resources does not succeed within the Online timeout interval. This implies that these resources will be failed over to another node only if the MonitorCommand for these resources returns Failed Offline.

**NoControl**
The NoControl option identifies the members of the equivalency as shadow resources. In this case, IBM Tivoli System Automation will not start or stop the members of the equivalency but will only react to changes of the OpState of the equivalency resource members. By default, IBM Tivoli System Automation will start and stop controllable resources within an equivalency, for example, of class IBM.Application.

## Attributes used by members of equivalencies

- A resource that is to be added to an equivalency *must* have this attribute:
  - OpState
- A resource that is to be added to an equivalency *may* have these attributes:
  - NodeNameList
  - ResourceType

For details of these attributes, refer to "Attributes used by resources" on page 26.

## Defining and administering equivalencies

### Making (creating) an equivalency

To make an equivalency among resources, you use the **mkequ** command.

For further details, see either the **mkequ** man page, or see the description of the **mkequ** command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

For example, to create an static equivalency called NetworkInterfaces of two ethernet interfaces eth0 located on Linux systems node01and node02 of the resource class IBM.NetworkInterface, you would enter:

```
mkequ NetworkInterfaces IBM.NetworkInterface:eth0:node01,eth0:node02
```

To create a dynamic equivalency called NetworkInterfacesDynamic containing all available ethernet interfaces in a cluster of Linux systems, you would enter:

```
mkequ -D "Name like 'eth%'" NetworkInterfacesDynamic IBM.NetworkInterface
```

In a cluster of AIX systems, you would enter:

```
mkequ -D "Name like 'en%'" NetworkInterfacesDynamic IBM.NetworkInterface
```

### List one or more equivalencies

To list one or more equivalencies, you use the **lsequ** command.

For further details, see either the **lsequ** man page, or the description of the lsequ command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

If you omit an equivalency name, *all* of the defined equivalencies will be listed. If you specify an equivalency, the persistent attributes of the this equivalency will be listed. If you specify the attribute name as operand, the attributes specified for the equivalency will be listed.

For example, to list the persistent attributes of the equivalency NetworkInterfaces, you would enter:

```
lsequ -A p -e NetworkInterfaces
```

## Changing an equivalency

To add, remove, or totally replace the resources contained in an equivalency, you use the **chequ** command.

For further details, see either the **chequ** man page or the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

You can also use this command to change the name of the equivalency.

For example, to add the resource eth1 located on system node01 that belongs to the resource class IBM.NetworkInterface, to an equivalency called NetworkInterfaces, you would enter:

```
chequ -u a NetworkInterfaces IBM.NetworkInterface:eth1:node01
```

## Removing an equivalency

To remove one or more equivalencies, you use the **rmequ** command.

For further details, see either the **rmequ** man page, or the description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

You specify one or more equivalencies using either the Equivalency name as operand, or selection string.

For example, to remove an equivalency called NetworkInterfaces, you would enter:

```
rmequ NetworkInterfaces
```

## Equivalencies

# Chapter 6. Using managed relationships

This chapter describes how you use managed relationships in these main sections:
- "What is a managed relationship?"
- "Attributes used by managed relationships" on page 54
- "Relationships for start / stop behavior" on page 56
- "Location relationships" on page 73
- "Creating and administering relationships" on page 84

**Note:** Note that System Automation decisions are based on a various factors, such as combinations of relationships between groups and resources, on attribute settings, and on the nominal states of groups. Each of these factors may have an impact on the automation behavior. The examples described below only illustrate the typical automation behavior that occurs when relationships are used, and the actual behavior may differ from the descriptions provided here.

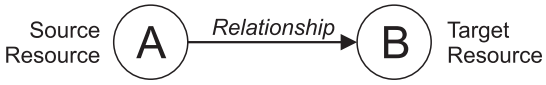
Table 4 lists the IBM Tivoli System Automation commands that you use for managed relationships. For details, see the command description in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

*Table 4. IBM Tivoli System Automation commands used with managed relationships*

| Command | Description |
|---|---|
| mkrel | Make a managed relationship |
| lsrel | List managed relationships |
| rmrel | Remove a managed relationship |
| chrel | Change a managed relationship |

## What is a managed relationship?

A managed relationship exists between a source resource, and one or more target resources. As a first example, during a resource group startup the source resource must be started after the target resource has become online: this example of a managed relationship uses a StartAfter value of the Relationship attribute.




As shown in the example, relationships always indicate a direction. Relationships can cross node boundaries.

As a second example, a source resource should if possible be started on the same node as the target resource: this example of a managed relationship uses an Affinity value of the Relationship attribute (described in "Relationship attribute" on page 55).

Relationship attributes may be used with Condition attributes (described in "Condition attribute" on page 55). As a third example, a source resource should, if possible, be started on the same node as the target resource, but only if the target

resource is online. This example of a managed relationship uses an Affinity value of the Relationship attribute, together with an IfOnline value of the Condition attribute.

The Relationship attribute is described in "Relationship attribute" on page 55, the Condition attribute is described in "Condition attribute" on page 55.

By using combinations of managed relationships, complex automation scenarios can be defined.

As mentioned above, a relationship is defined between a source and one or more target resources. If you remove the source resource of a relationship (can be a resource group member or the underlying RMC resource), the relationship is deleted. If you remove the last target resource from the list of target resources (can be a resource group member or the underlying RMC resource), the relationship is not deleted. You must remove this relationship with the **rmrel** command.

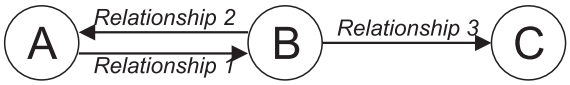
The reason for this behavior is that no relationships should accidentally be deleted. There was a reason why, for example, a DependsOn was defined from a source resource to a target resource. The source resource cannot function properly without the target resource. So the relationship should not be deleted automatically unless you tell IBM Tivoli System Automation to do so.

Managed relationships are supplied in IBM Tivoli System Automation resource class IBM.ManagedRelationship.

## Attributes used by managed relationships

**Related Sections:**
- "Attributes used by resources" on page 26
- "Attributes used by resource groups" on page 34
- "Attributes used by equivalencies" on page 49

The following picture shows another example of a managed relationship:




A managed relationship has the attributes described in the sections below.

### Name attribute

You use the Name persistent attribute to specify the name you wish to use for the managed relationship. This attribute is optional. It makes it easier to change or delete relationships.

### Source attribute

You use the Source persistent attribute to specify the source resource of the managed relationship.

### Target attribute

You use the Target persistent attribute to specify the list of target resources of the managed relationship.

## Relationship attribute

You use the Relationship persistent attribute to specify the relationship that is to be applied between source and target resources. There are two types of relationships, start / stop dependencies and location dependencies:
**Start / Stop dependencies**:
- StartAfter
- StopAfter
- DependsOn
- DependsOnAny
- ForcedDownBy

Start / Stop dependencies are used to define a start / stop behavior.
**Location dependencies**
- Collocated
- AntiCollocated
- Affinity
- AntiAffinity
- IsStartable

Location dependencies are used for locating resources on nodes.

## Condition attribute

The Condition persistent attribute specifies a condition to be used together with all Location relationships (described in "Location relationships" on page 73), *except for* the IsStartable managed relationship. The IfPossible condition can only be used in combination with the StartAfter relationship.

The Condition persistent attribute defines when the relationship is considered applicable. These are the conditions that can be applied:

- IfOnline
- IfNotOnline
- IfOffline
- IfNotOffline
- IfPossible
- None

## Relationships for start / stop behavior

IBM Tivoli System Automation provides the following relationships which can be used to define a start/stop behavior:

- **StartAfter**
- **StopAfter**
- **DependsOn**
- **DependsOnAny**
- **ForcedDownBy**

The source of a start/stop relationship is either a member of a resource group or a resource group. See "What is a resource group?" on page 31 for more information about resource groups.

The target of a start/stop relationship is either

- a member of a resource group or a resource group.
- an equivalency.
- an RSCT resource (which is not a managed resource) which has to provide an OpState attribute.

Note that in case of a DependsOn relationship and source or target resources or both being groups, these groups must have a member location of collocated.

A start command cannot be issued against a resource directly. Therefore you start a resource by setting the nominal state of the resource group of which resource is a member to online.

### StartAfter relationship

Use the StartAfter relationship to ensure that the source resource is only started when the target resource(s) are online.

The StartAfter relationship provides the following behavior scheme:


- With the start behavior StartAfter defines a start sequencing for resources A and B:
  When source resource A has to be started, then the target resource B is started first. After resource B has become online, resource A is started.
  Note that resource A and resource B can be started on different nodes.

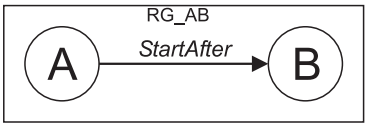
  Using the IfPossible condition with StartAfter relationship between two resource groups induces a variant of this behavior, because the target resource group B is bypassed if it cannot be bound and ends up in a sacrificed binding state (see "Location relationship resolution: Binding algorithm" on page 87). In this case, the source of the relationship is started ignoring the relationship.

The StartAfter relationship does not provide a force down behavior (see "DependsOn relationship" on page 64).

## Details on the start behavior of the StartAfter relationship

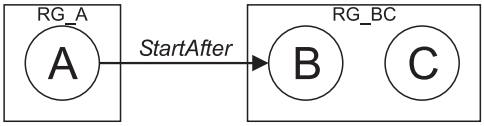
The start behavior is controlled via the operational state (OpState) of the target resource. At the time when the operational state of resource B has become online, resource A is started.

In many cases resources A and B are members of the same resource group.



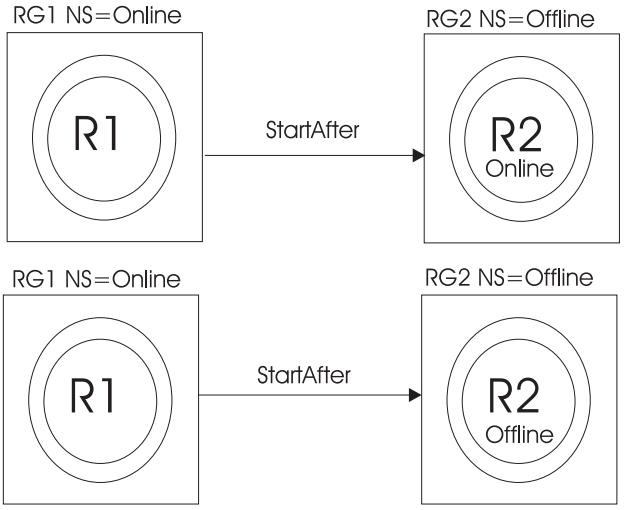

Setting their resource group's nominal state to online causes that both members A and B are started. Due to the StartAfter relationship from A to B resource B is started first. When the operational state of resource B is online, resource A is started.

If resource A is a member of resource group RG_A, and resource B is a member of resource group RG_BC, and a StartAfter relationship is defined between A and B. Then the start behavior of the StartAfter relationship is triggered by setting the nominal state of RG_A to online.

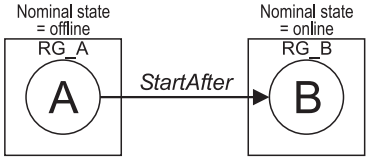



Due to the start sequence of the StartAfter relationship resource B has to be started first. In case RG_BC's nominal state is set to offline, the following conflict exists: RG_BC wants resource B to be offline whereas the StartAfter relationship forces B to be started. IBM Tivoli System Automation resolves this conflict in such a way that the online request is always more important than the offline request. Therefore resource B is started even though other possible group members of RG_BC will not be started since the nominal state of their group is offline. After resource B is online, IBM Tivoli System Automation will start resource A. Resource C is not started.

If resource A was started, and resource B is started due to a StartAfter relationship (but the nominal state of the comprising group remains Offline), then resource B is not recovered, which means that B is not restarted in case of failure. If the StartAfter relationship was created between the resource groups RG_A and RG_BC, the target group members would be restarted in case of a failure.

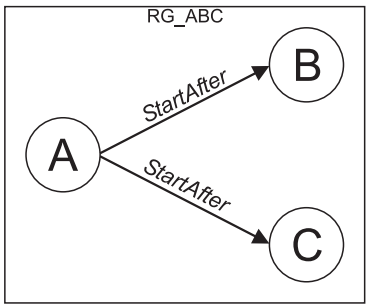



The start order only acts in the forward direction of the relationship. In case resource A and resource B are part of different resource groups (A belongs to RG_A and B belongs to RG_B), then setting the nominal state of RG_B to online does not cause any action on resource A since resource B has no forward relationship to resource A.



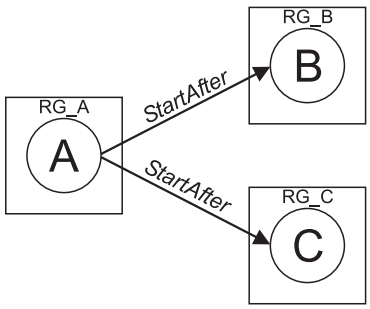

When RG_A's nominal state is set to online, the resource A can be started right away since resource B is already online.

In another scenario it also might be the case that resource A has a StartAfter relationship to resource B and resource C.




In this case starting A requires that both resources B and C are online before IBM Tivoli System Automation can start resource A. For instance A, B, and C are members of the resource group RG_ABC. Setting the nominal state of RG_ABC to online causes that resources B and C are started in parallel first. When the operational state of both resources is online, then resource A is started.

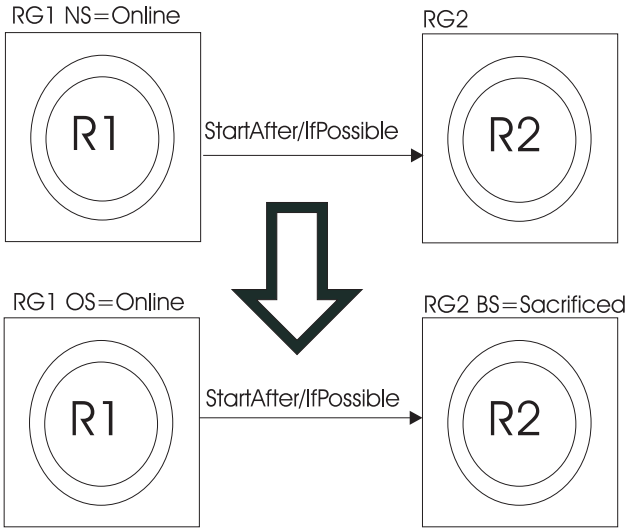
It is also possible that resource A is member of resource group RG_A, resource B is member of resource group RG_B, and resource C is member of resource group RG_C.




A has a StartAfter relationship to both B and C. Setting RG_A's nominal state to online causes that due to the StartAfter relationship resource C and resource B are started. After both resources B and C are online, A is started.

The example depicted below shows the start behavior that occurs when two resource groups ("RG1" and "RG2") are connected through a StartAfter/IfPossible relationship and the target resource group ("RG2") cannot be bound:

- The source of the relationship ("RG1") is started.
- The relationship to the target resource group ("RG2") is not honored and its bindingState is set to Sacrificed. Note that a different behavior occurs if both resource groups are members of another group.

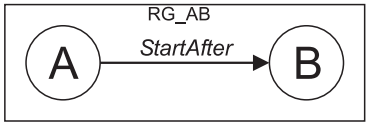



**Using the IfPossible condition with StartAfter relationships:** You can specify the IfPossible condition with StartAfter relationships. The condition specifies that the target resource group may be bypassed if it cannot be bound. In such a case, the target resource group ends up in state Sacrificed and the StartAfter relationship is ignored.

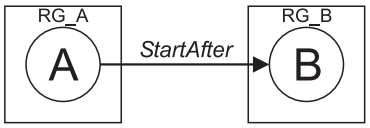
## Details on the stop behavior of the StartAfter relationship

Target resource B cannot be stopped while source resource A is online. If the NominalState attribute of source resource A is changed to Offline, target resource B automatically stops. Both resources can be simultaneously stopped.

In many cases source resource A and target resource B are members of the same resource group. So their NominalState values are identical.

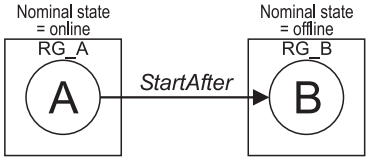



Set the NominalState attribute of the RG_AB to offline to stop both members A and B. Since the StartAfter relationship does not require a stop sequence, resources A and B can be simultaneously stopped.



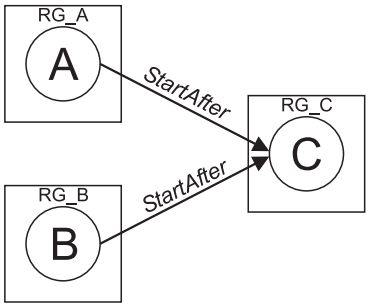

Provided that RG_B has a NominalState of Online, you can start and stop RG_A without affecting resource group RG_B. It remains online.
If you set the NominalState of RG_B to Offline and set the NominalState of RG_A to Online, target resource B will start before source resource A.
If you set the NominalState of RG_A to Offline, then resources A and B are simultaneously stopped.

Consider the following stop behavior:




Provided the NominalState of RG_A is Online and the NominalState of RG_B is Offline, resource A and resource B are online. Now set the NominalState of RG_A to Offline. Resource A and resource B simultaneously stop. The reason for this behavior is that resource B started due to the start request on resource group RG_A which was passed on via the StartAfter relationship. Setting RG_A to Offline removes the start request, and the NominalState of Offline from resource group RG_B causes resource B to be stopped.
The StartAfter relationship causes the typical stop behavior: Resource A and B can be simultaneously stopped.

Resources A, B, and C are members of individual resource groups RG_A, RG_B, and RG_C.



Resource C must be online to support both resources A and B. As long as the NominalState of either or both RG_A and RG_B is online, resource C must be kept Online, even if RG_C's NominalState is Offline. Only when both RG_A and RG_B have a NominalState of Offline, resource C can be stopped. This will be the case if RG_C's NominalState is Offline, too.

## Rules for using the StartAfter relationship

1. The StartAfter relationship must not conflict with an existing DependsOn relationship.
2. The StartAfter relationship does not assume that a Location relationship exists between managed resources. If you wish to define a Location relationship (see "Location relationships" on page 73), you must create an additional relationship for this purpose.
3. If IBM Tivoli System Automation is requested to start the source resource, it will however always attempt to first start the target resource.
4. If the target resource fails, this does not mean that the source resource will then be stopped.

## StopAfter relationship

Use the StopAfter relationship to ensure that the source resource can only be stopped when the target resource has been already stopped.

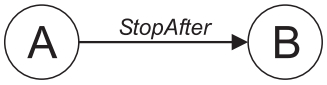
The StopAfter relationship provides the following behavior scheme:


- Resource A will not be stopped unless the target resource has been brought Offline before (including Failed Offline).

The StopAfter relationship does not provide a start and a force down behaviour (see “StartAfter relationship” on page 56 and “DependsOn relationship” on page 64).

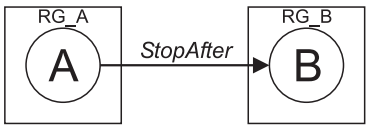
### Details on the stop behavior of the StopAfter relationship

Source resource A cannot be stopped while target resource B is Online. If the OpState attribute of target resource B changes to Offline or Failed Offline, source resource A automatically stops



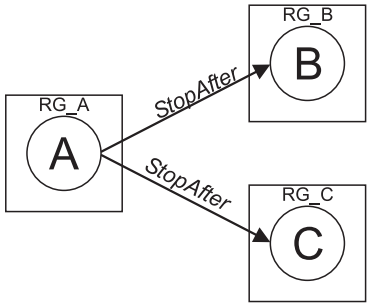

In many cases source resource A and target resource B are members of the same resource group. Set the NominalState attribute of the RG_AB to Online to start both members A and B. Since the StopAfter relationship does not require a start sequence, resources A and B can be simultaneously started. Setting their resource group’s NominalState attribute to Offline causes that members are stopped. Due to the relationship from A to B, resource B is stopped first. When the operational state of resource B is Offline, resource A is stopped.


In case resource A and B are part of different resource groups (A belongs to RG_A, and B belongs to RG_B) and RG_B has a NominalState of Offline, you can start and stop RG_A without any dependency to resource group RG_B. If you set the NominalState of RG_B to Online and set the NominalState of RG_A to Offline, source resource A cannot stop as long as target resource B is Online.




If the NominalState of RG_A is Offline, you can start or stop RG_B without any dependency to resource A.
It is also possible that resource A is a member of resource group RG_A, resource B

is a member of resource group RG_B, and resource C is a member of resource group RG_C. A has a StopAfter relationship to both B and C.



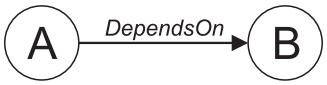

If the NominalState of RG_A is Online and you want to stop it, RG_A cannot be stopped as long as the NominalState of both RG_B and RG_C is Online. Only when both RG_B and RG_C have a NominalState of Offline or Failed Offline, resource A can be stopped.

## DependsOn relationship

IBM Tivoli System Automation uses the DependsOn relationship to ensure that the source resource can only be started when the target resource(s) is online. It is used in a similar way to the StartAfter relationship, except:

- A DependsOn relationship also includes an implicit collocation (explained in "Collocated relationship" on page 75) between the source and target resources.
- If a target resource fails, the source resource will also be stopped.

The DependsOn relationship provides the following three behavior schemes:




Resource A depends on the functionality of resource B, and this means that resource A cannot function without resource B. Therefore the force down behavior is introduced (see item 3 in the following list).

1. With the start behavior DependsOn defines a start sequencing for resources A and B with an implicit collocation:
   When resource A (source) has to be started then the target resource B is started first. After resource B has become online, resource A (source) is started on the same node.
2. With the stop behavior DependsOn defines a stop sequence for resource A and B:
   When resource B (target) has to be stopped, then source resource A is stopped first. After resource A has become offline, resource B (target) is stopped
3. Force down behavior in case the target resource fails: When target resource B has failed resource A is also stopped. Then a restart is triggered according to the start behavior described in 1.

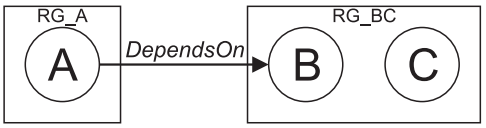
### Details on the start behavior of the DependsOn relationship

The start sequencing of the DependsOn relationship is controlled via the operational state (OpState) of the target resource. At the time when the operational state of resource B has become online, resource A is started. In addition to the start sequence, DependsOn provides a collocated constraint which causes that resource A has to be started on the same node where resource B was started. Therefore resource B is already started on a node where resource A can be started afterwards. The collocated constraint which is part of the DependsOn relationship corresponds to the behavior of the collocated relationship. For further details on this behavior see "Collocated relationship" on page 75.

In many cases resource A and resource B are members of the same resource group.

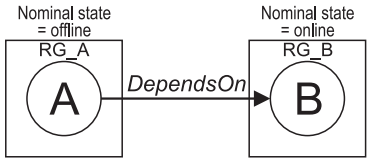

Setting their resource group's nominal state to online causes that both members A and B are started. Due to the DependsOn relationship from A to B resource B is started first. When resource B's operational state is online, resource A is started on the same node.

If resource A is a member of resource group RG_A, and resource B is a member of resource group RG_BC, and a DependsOn relationship is defined from A to B, then the start behavior of the DependsOn relationship is triggered by setting the nominal state of RG_A to online.



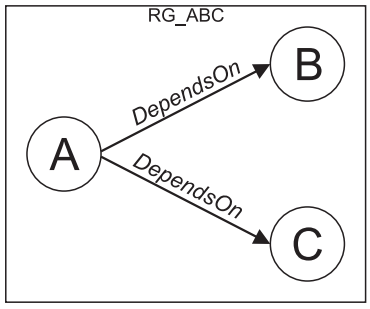

Due to the start sequence of the DependsOn relationship, resource B has to be started first. In case RG_BC's nominal state is set to offline there is the following conflict: RG_BC wants resource B to be offline whereas the DependsOn relationship forces B to be started. IBM Tivoli System Automation resolves the conflict in such a way that the online request is always more important than the offline request. Therefore resource B is started even though other possible group members of RG_BC will not be started as their group's nominal state is offline. After resource B is online, IBM Tivoli System Automation will start resource A. Of course the resources A and B are started on the same node. Resource C is not started.

The start order only takes effect in the forward direction of the relationship. In case resource A and resource B are part of different resource groups (A belongs to RG_A and B belongs to RG_B),




then setting the nominal state of RG_B to online does not cause any action on resource A as resource B has no forward relationship to resource A. When RG_A's nominal state is then also set to online, the resource A can be started right away on the same node as resource B is already online

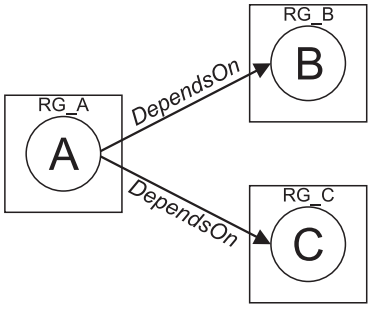
In another scenario it also might be the case that resource A has a DependsOn relationship to resource B and resource C.




In this case starting A requires that both resources B and C are online before IBM Tivoli System Automation can start resource A. For instance A, B, and C are members of the resource group RG_ABC. Setting the nominal state of RG_ABC to

online causes that resources B and C are started in parallel first. When both resources' operational state is online then resource A is started. All three resources are started on the same node as A has to be started on the same node where B and C are running .

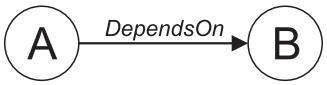
It is also possible that resource A is member of resource group RG_A, resource B is member of resource group RG_B and resource C is member of resource group RG_C.




A has a DependsOn relationship to both B and C. Setting RG_A's nominal state to online causes resource B and resource C to be started. After both resources B and C are online, A is started on the same node.

### Details on the stop behavior of the DependsOn relationship

You can control the stop sequencing of the DependsOn relationship via the operational state (OpState) of the source resource.




When the resource A's OpState has become offline, then resource B can be stopped.

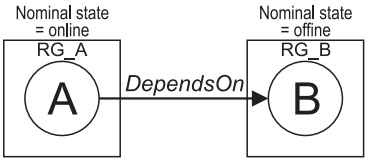
In many cases resource A and resource B are members of the same resource group.


Set the nominal state attribute of the resource group to offline to stop both members A and B. Due to the DependsOn relationship resource A is stopped first. When resource A is offline resource B is stopped

Resource A is a member of resource group RG_A, and resource B is a member of resource group RG_B, and a DependsOn relationship is defined from A to B. You can trigger the stop behavior of the DependsOn relationship by setting the nominal state of RG_B to offline (stopping resources directly is not possible in IBM Tivoli System Automation). Due to the DependsOn relationship resource A should stop first. There is a conflict if the nominal state of resource group A is set to online: RG_A wants resource A to be online whereas the DependsOn relationship causes it to be stopped.

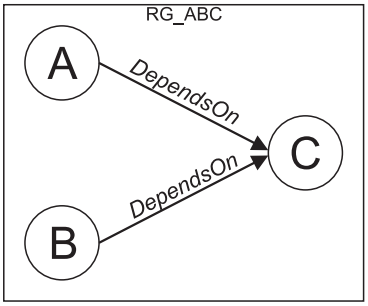

This conflict is resolved in such a way that in IBM Tivoli System Automation always the online request is more important than the offline request. Therefore resource A is kept online and resource B cannot be stopped. Only if the nominal state of RG_A is set to offline, resource A can be stopped. When resource A is offline, resource B is stopped afterwards.

There is also an implicit stop behavior to consider:

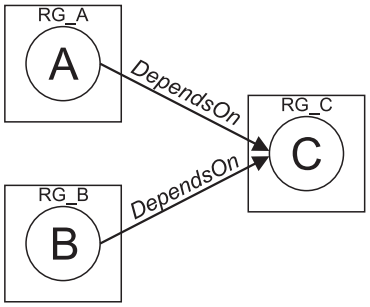



When the nominal state of RG_A is online, and the nominal state of resource group RG_B is set to offline, then as described above in the start scenario resource A and B are online. Now the nominal state of RG_A is set to offline. This causes resource A to stop. In addition, resource B will be stopped. The reason for this is that it was started due to the start request on resource group RG_A which was propagated via the DependsOn relationship to resource B. Since this resource group RG_A is set to offline, the start request is removed and the nominal state of offline from resource group RG_B causes B to be stopped. The DependsOn relationship causes the typical stop behavior: Resource B cannot be stopped before resource A is stopped. Therefore resource A is stopped first. When A is offline, then resource B is stopped.

Another scenario is that resource A and B have a DependsOn relationship to C. Stopping resource C requires that both resources A and B are brought offline first.




For instance A, B, and C are members of the same resource group RG_ABC. Setting the nominal state of RG_ABC to offline causes that resources A and B are stopped first. When the operational state of both resources is offline, then resource C is stopped. An alternative example is that resources A, B, and C are members of individual resource groups RG_A, RG_B, and RG_C, respectively.

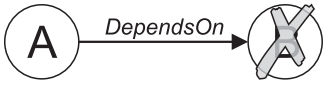



Setting the nominal state of RG_C to offline triggers the stop behavior of the DependsOn relationship. Here the nominal state of the resource groups RG_A and RG_B might overrule the stop behavior. As long as the nominal state of RG_A or RG_B is online, resource C cannot be stopped. The reason for this is that in conflict situations an online request always overrules an offline request. Therefore the stop behavior of the DependsOn relationship is defered until the nominal state of RG_A and RG_B is set to offline. When their members A and B are offline, then resource C is also stopped.

## Details on the force down behavior of the DependsOn relationship

The basic principle of the DependsOn relationship is that source A depends on the functionality of the target resource B. When the target resource B fails then the source resource A cannot function anymore. Therefore, it is not sufficient to restart B. Due to a failure of B resource A will also be forced down. And then both resources will be restarted according to the start behavior: First B, then A.

As example one can define a resource A which has a DependsOn relationship to resource B.

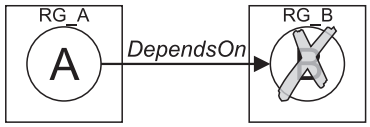



Both resources are online. In case resource B fails, resource A stops and then the normal start behavior occurs. Resource B will be restarted, and then resource A will be started.

Resource A and resource B are members of the same resource group RG_AB.

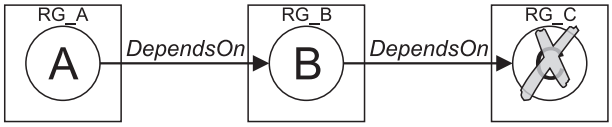

In addition, the relationship resource A DependsOn resource B is defined. When RG_AB is set to online, resource B is started first, then resource A is started. In case resource B fails or goes offline, resource A is also stopped. Afterwards a normal restart is performed with the start sequence of DependsOn:
B is started before A is started.

It could also be the case that resource A is member of resource group RG_A, and resource B is member of resource group RG_B, and A has a DependsOn relationship with resource B.




When resource group RG_A is set to online and RG_B's nominal state is offline, then resource B is started first and resource A is started afterwards. The force down behavior of the DependsOn relationship is triggered by a failure of resource B. It causes that resource A will also be stopped. This will occur even though RG_A's nominal state is online. In IBM Tivoli System Automation such a conflict is always resolved in such a way that a force down behavior is always more important than the online request of a resource group.

The force down behavior is propagated through chains of DependsOn relationship. Given the following scenario: resource A is member of resource group RG_A, Resource B is member of resource group RG_B, and resource C is member of resource group RG_C with the relationships A DependsOn B and B DependsOn C.




Let's assume that resource group RG_A is set to online which causes that the three resources C, B, and A were sequentially started and are in online state. Now resource C fails. This causes resources A and B to be forced down: First, resource A is stopped, then resource B is stopped. The reason is that the force down behavior has a higher importance than a normal online request.

## Rule for using the DependsOn relationship

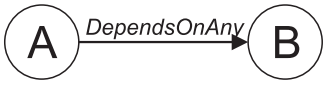
There is one rule for using the DependsOn relationship:

1. If the source or target resource is a group, all members of the group must be collocated.

## DependsOnAny relationship

The behavior of the DependsOnAny relationship is identical to the DependsOn relationship except that it does not provide the collocated constraint for the start sequence. Therefore source and target resources can be started either on the same node or on different nodes.

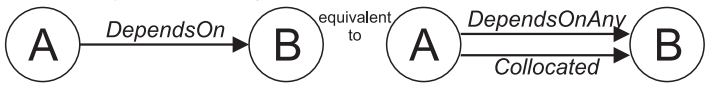
The DependsOnAny relationship provides the following three behavior schemes:




1. With the start behavior DependsOnAny defines a start sequencing for resource A and B **without** a location relationship:
   When resource A (source) has to be started, then the target resource B is started first. After resource B has become online, resource A (source) is started. Note that the only difference to the DependsOn relationship is that Resource A and resource B can be started on different nodes.
2. With the stop behavior DependsOnAny defines a stop sequence for resource A and B:
   When resource B (target) has to be stopped, then source resource A is stopped first. After resource A has become offline, resource B (target) is stopped.
3. Force down behavior in case the target resource fails: When target resource B has failed resource A is also stopped. Then a restart is triggered according to the start behavior described in 1.

See DependsOn relationship for further details on the DependsOnAny relationship.

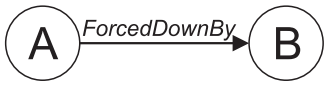
**Note:** The scenario ***A ---> DependsOn ---->B*** corresponds to the scenario ***A ---> DependsOnAny ---> B*** and ***A----> Collocated----->B***



## ForcedDownBy relationship

Use the ForcedDownBy relationship to ensure that the source resource will be brought down if the target resource comes offline.

The ForcedDownBy relationship provides the following behavior scheme:



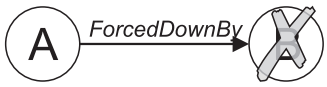

- Resource A must be forced Offline in the event that either the target resource goes Offline unexpectedly or the target resource itself is forced Offline. The stop of resources A and B can happen in parallel. The force down of resource A will be triggered when resource B enters any of the regular down states (Offline) after having previously been in an Online state or in any of the terminal down states (Failed Offline), regardless of its former state.

The ForcedDownBy relationship does not provide a start and a stop behaviour (see "StartAfter relationship" on page 56, "StopAfter relationship" on page 62, and "DependsOn relationship" on page 64).

### Details on the force down behavior of the ForcedDownBy relationship

The basic principle of the ForcedDownBy relationship is that source A must be forced Offline when target resource B goes Offline or fails.

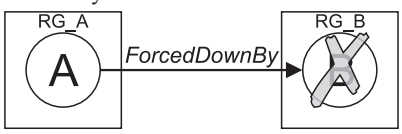
As example one can define a resource A which has a ForcedDownBy relationship to resource B.



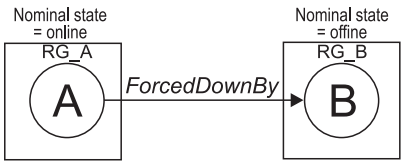

Both resources are Online. In case resource B is stopped or fails, resource A will be forced down.

It could also be the case that resource A is member of resource group RG_A, and resource B is member of resource group RG_B, and A has a ForcedDownBy relationship with resource B.
When the NominalState attribute of the resource groups RG_A and RG_B is set to Online, then resource A and B are started without any dependencies on each other. The force down behavior of the ForcedDownBy relationship is triggered either:

1. By a failure of resource B. It causes that resource A will be stopped, too. This will occur even though RG_A's nominal state is Online. But since the nominal state of RG_A is still Online in this case, resource A will be restarted by IBM Tivoli System Automation.




2. Or by a stopping of resource B.



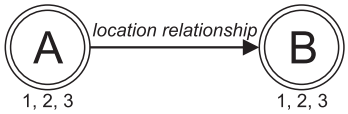
Setting the nominal state of RG_B to Offline causes that resource A will be stopped, too. This will occur even though RG_A's nominal state is Online. But since the nominal state of RG_A is still Online in this case, resource A will be restarted by IBM Tivoli System Automation.

## Location relationships

IBM Tivoli System Automation provides the following relationships which can be used to define location relationships:

- **Collocated**
- **AntiCollocated**
- **Affinity**
- **AntiAffinity**
- **IsStartable**

For example, resources A and B are floating resources which can be started on node1, node2, and node3:




The idea behind these relationships is to define location constraints between resources. Resource types like floating resources, and groups provide a list of nodes on which these can be started. Resource A and resource B are floating resources which can be started on node1, node2, and node3.

A requirement could be that resource A must always be started on the node where resource B is already running or supposed to be running. This behavior can be specified by defining a Collocated relationship from A to B.

The opposite behavior which requires that resource A must not be started on the node where resource B is already running can be specified by defining the AntiCollocated relationship.
In case of the requirement that resource A should - if possible - be started on the node where resource B is running, otherwise it can be started elsewhere, the Affinity relationship is used. Compared to the Collocated relationship, the Affinity relationship has 'soft' location relationships.

The AntiAffinity relationship is used to define that resource A should not be started - if possible - where B is already running. Only if this requirement cannot be satisfied, process A can be started on the node where B is located. Like the Affinity relationship also the AntiAffinity relationship has 'soft' location constraints compared to the AntiCollocated relationship.

The IsStartable relationship defines that source resource A can only be placed on a node where target resource B is startable. This relationship is only considered if the source and target resources have nominal state online. When one of the resources (source or target) does not have a nominal state of online, the IsStartable relationship will be discarded together with the resources that have a nominal state of Offline.

## Conditions IfOnline, IfOffline, IfNotOnline, and IfNotOffline

You can specify the following conditions together with all location relationships except IsStartable. These conditions are:

**IfOnline** IfOnline defines that a location relationship is only evaluated when the target resource's OpState is Online. Otherwise the location is ignored. IfOnline does not include states such as Pending Online, and Pending Offline.

**IfOffline** IfOffline means that a location relationship is only evaluated when the target resource's OpState is either Offline or Failed Offline or Unknown. Otherwise the location relationship is ignored.

**IfNotOnline** IfNotOnline means that a location relationship is only evaluated when the target resource is **not** in an Online state. IfNotOnline includes states such as Pending Online and Stuck Online. Otherwise the location relationship is ignored.

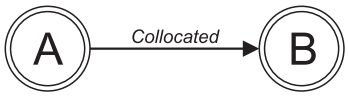
**IfNotOffline** IfNotOffline means that a location relationship is only evaluated when the target resource is **not** in an Offline or Failed Offline or Unknown state. Otherwise the location relationship is ignored.

## Rules for using location relationships

1. The source of a location relationship is either a member of a resource group or a resource group. See "What is a resource group?" on page 31 for more information about resource groups.
2. The target of a location relationship is either
   - a member of a resource group or a resource group.
   - an RMC resource (which is not a managed resource) which has to provide a start /stop method and an OpState attribute.
3. If the source or target resource is a group, all members of the group must be collocated.

## Collocated relationship

IBM Tivoli System Automation uses the Collocated relationship to ensure that the source resource and its target resource are located on the *same* node. The Collocated relationship provides the following behavior scheme:



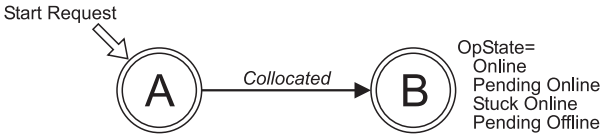

- The Collocated relationship defines that on start of resource A it can only be started on the node where resource B is already running.

The Collocated relationship can be used together with a Condition attribute as described on page 76.

### Details on the principle behavior of the Collocated relationship

The following describes in detail four states the Collocated relationship can take:

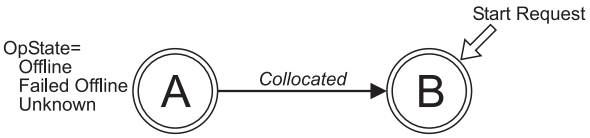
**Case I:**
On start of resource A place it on the same node where resource B is already running . 'Running' means that the OpState of resource B is either Online, Pending Online, Stuck Online, or Pending Offline.




This behavior represents the standard situation.

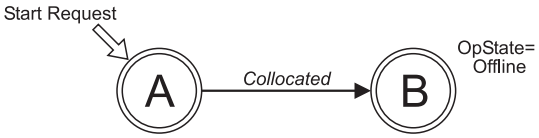
The collocated relationship tries to optimize the node selection based on predictions for future situations. Here the following cases are possible:

**Case II:**
Resource B is started and resource A is in an Offline, Failed Offline or Unknown state.




Generally you would expect that the node selection for resource B is independent of resource A. But when IBM Tivoli System Automation selects a node for resource B, then a node is selected on which resource A could also be started in future. The reason for this prediction approach is that it later simplifies the start behavior for resource A: if no error situation occurs it is ensured that after resource B was started, resource A can be started on the same node where resource B runs.

**Case III:**
Resource A is started and resource B is in an offline state.



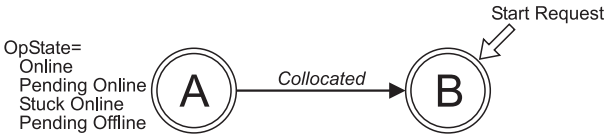

Theoretically resource A could now be placed on any of the nodes of its node list since A cannot be bound to a node on which B is running. Here again the prediction approach tries to find a node location for resource A where resource B

could also be started in future. Therefore, IBM Tivoli System Automation determines the same node location for both resources A and B even though it will only start resource A. The internal IBM Tivoli System Automation behavior works as follows: When resource A has to be started IBM Tivoli System Automation determines a node location for both resources A and B, and then starts resource A. (Note: the start of resource B is not driven by the collocated relationship. This is done by another start/stop relationship or a group behavior).

A summary of the prediction approach is: If either resource A or resource B is started, and the other resource is in an offline state, then IBM Tivoli System Automation determines a node location where both resources A and B are logically bound before one of them is started.

Note that the optimization for the node location is just a prediction based on the current circumstances. The prerequisites the decision of the node selection was based upon can change over time.
A scenario of an erroneous prediction of a node selection could be the following: Resources A and B are floating resources and can be located on node 1, 2, 3. The relationship A -- Collocated ---> B is defined. Now resource B has to be started. Due to the collocated relationship IBM Tivoli System Automation could select node1 for resources A and B. Then resource B is started. After a while an administrator usage error causes that resource A cannot be started on node 1 anymore. The OpState of the resource on node 1 is FailedOffline. Then a request causes that resource A has to be started. Since resource A cannot be started on node 1 anymore, a conflict situation occurs which has to be solved as described later.

**Case IV:**
Another possible state could be that resource A is already in a running state (OpState is either Online, PendingOnline, StuckOnline, or PendingOffline) when resource B is started.




At the time when resoure A was started resource B already got the same node selected. If no error occurred resource B can be started there. If there was a problem which prevented resource B from starting on the previous selected node, the resource gets unbound and at start time of resource B a new node location has to be found. This means that resource B can be started on another node.

The following relationships with conditions can be defined:

- **Collocated/IfOnline**

  The relationship A ---> Collocated/IfOnline -----> B means that the location relationship is only considered when resource B is in a online state. Otherwise the location relationship is ignored. IfOnline does not include states such as Pending Online, and Pending Offline.
- **Collocated/IfOffline**

  The relationship A ---> Collocated/IfOffline -----> B means that the location relationship is only valid when resource B is in an Offline, Failed Offline or Unknown state.
- **Collocated/IfNotOnline**

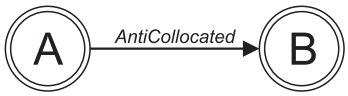
The relationship A ---> Collocated/IfNotOnline -----> B means that the location relationship is only valid, when resource B is not in an online state.

- **Collocated/IfNotOffline**

  The relationship A ---> Collocated/IfNotOffline -----> B means that the location relationship is only valid, when resource B is not in an Offline, Failed Offline or Unknown state.

## AntiCollocated relationship

IBM Tivoli System Automation uses the AntiCollocated relationship to ensure that the source resource and its target resource are located on *different* nodes. The AntiCollocated relationship provides the following behavior scheme:




- TheAntiCollocated relationship defines that on start of resource A it can only be started on a different node where resource B is already running.

The AntiCollocated relationship can be used together with the Condition attribute as described on page 79.

### Details on the principle behavior of the AntiCollocated relationship

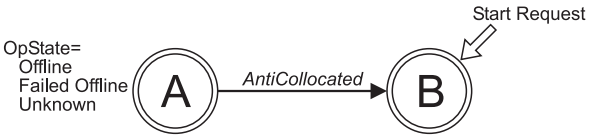
The following describes in detail four states the AntiCollocated relationship can take:

**Case I:**
On start of resource A place it on a different node than the one where resource B is currently running. 'Running' means that the OpState of resource B is either Online, Pending Online, Stuck Online, or Pending Offline.


This behavior represents the standard situation.

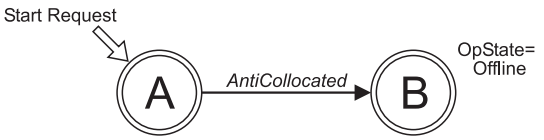
The AntiCollocated relationship tries to optimize the node selection based on predictions for future situations. Here the following cases are possible:

**Case II:**
Resource B is started and resource A is in an Offline, Failed Offline or Unknown state.




Generally you would expect that the node selection for resource B is independent from resource A. But when IBM Tivoli System Automation selects a node for resource B, then a node is selected which allows that resource A can be started on another node in the future. The reason for this prediction approach is that it later simplifies the start behavior for resource A: if no error situation occurs it is ensured that after resource B was started, resource A can be started on another node where resource B is not running. This corresponds to the description of **Case I** .

**Case III:**
Resource A is started and resource B is in an offline state (Offline, Failed Offline).



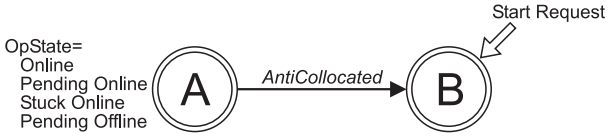

Theoretically resource A could now be placed on any of the nodes of its node list.

Here again the prediction approach tries to find a node location for resource A that allows resource B to be started on another node in the future. Therefore IBM Tivoli System Automation determines a node location for resource B even though it will only start resource A.

Summary of the prediction approach is:
If resource A is in an offline state and either resource A or resource B is started (see Case II and Case III), then IBM Tivoli System Automation determines a different node location for both resources A and B before one of them is started

As already mentioned in the description of the Collocated relationship it may happen that the prediction based on the current circumstances might be wrong over time. Nevertheless the prediction approach will simplify the automation behavior in most cases.

**Case IV**:
Resource A is already in a running state (OpState is either Online, Pending Online, Stuck Online, or Pending Offline) when resource B is started.




At the time when resoure A was started (see Case III) resource B already got another node selected. If no error occurred, resource B can be started there. If there was a problem which caused that resource B cannot be started anymore on its previous selected node, at start time of resource B a new node location is found. This means that resource B can be started anywhere, even where resource A is already running.

The following relationships with conditions can be defined:

- **AntiCollocated/IfOnline**

  The relationship A ---> AntiCollocated/IfOnline -----> B means that the location relationship is only valid, when resource B is in a online state. Otherwise the location relationship is ignored. IfOnline does not include states such as Pending Online, and Pending Offline.
- **AntiCollocated/IfOffline**

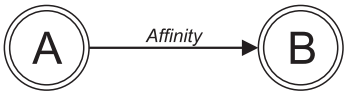
  The relationship A ---> AntiCollocated/IfOffline -----> B means that the location relationship is only valid when resource B is in an Offline, Failed Offline or Unknown state.
- **AntiCollocated/IfNotOnline**

  The relationship A --->AntiCollocated/IfNotOnline -----> B means that the location relationship is only valid, when resource B is not in an online state.
- **AntiCollocated/IfNotOffline**

  The relationship A ---> AntiCollocated/IfNotOffline -----> B means that the location relationship is only valid, when resource B is not in an Offline, Failed Offline or Unknown state.

## Affinity relationship

The Affinity relationship provides the following behavior scheme:




- The Affinity relationship defines that on start of resource A the same node is chosen where resource B is already running, if possible. If other location relationships are inhibiting this, resource A can also run on another node.

The Affinity relationship is very similar to the Collocated relationship. Therefore the Affinity relationship defines a soft location relationship whereas the Collocated relationship is a hard location relationship.

The Affinity relationship can be used together with the Condition attribute (described in "Condition attribute" on page 55).

The following relationships with conditions can be defined:

- **Affinity/IfOnline**

  The relationship A ---> Affinity/IfOnline -----> B means that the location relationship may only be considered when resource B is in a online state. Otherwise the location relationship is ignored. IfOnline does not include states such as Pending Online, and Pending Offline.
- **Affinity/IfOffline**

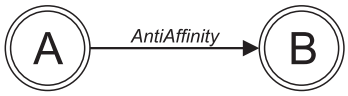
  The relationship A --->Affinity/IfOffline -----> B means that the location relationship may be only valid when resource B is in an Offline, Failed Offline or Unknown state.
- **Affinity/IfNotOnline**

  The relationship A ---> Affinity/IfNotOnline -----> B means that the location relationship may only be valid, when resource B is not in an online state.
- **Affinity/IfNotOffline**

  The relationship A ---> Affinity/IfNotOffline -----> B means that the location relationship may only be valid, when resource B is not in an Offline, Failed Offline or Unknown state.

## AntiAffinity relationship

The AntiAffinity relationship provides the following behavior scheme:




- The AntiAffinity relationship defines that on start of resource A a different node than the one where resource B is already running is chosen, if possible. If other location relationships are inhibiting this, resource A can also run on the same node.

The AntiAffinity relationship is very similar to the AntiCollocated relationship. Therefore the AntiAffinity relationship defines a soft location relationship whereas the AntiCollocated relationship is a hard location relationship.

The AntiAffinity relationship can be used together with the Condition attribute (described in "Condition attribute" on page 55).

See also "Location relationships" on page 73.

The following relationships with conditions can be defined:

- **AntiAffinity/IfOnline**

  The relationship A ---> AntiAffinity/IfOnline -----> B means that the location relationship may only be valid, when resource B is in a online state. Otherwise the location relationship is ignored. IfOnline does not include states such as Pending Online, and Pending Offline.
- **AntiAffinity/IfOffline**

  The relationship A ---> AntiAffinity/IfOffline -----> B means that the location relationship may only be valid when resource B is in an Offline, Failed Offline or Unknown state.
- **AntiAffinity/IfNotOnline**

  The relationship A ---> AntiAffinity/IfNotOnline -----> B means that the location relationship may only be valid, when resource B is not in an online state.
- **AntiAffinity/IfNotOffline**

  The relationship A ---> AntiAffinity/IfNotOffline -----> B means that the location relationship may only be valid, when resource B is not in an Offline, Failed Offline or Unknown state.

## IsStartable relationship

The IsStartable relationship provides the following behavior scheme:


- The IsStartable relationship defines that resource A can only be placed on a node where resource B is startable when the resources A and B have a nominal state of online.

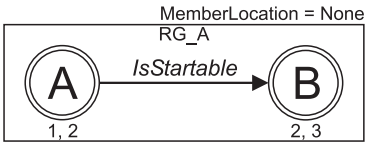
IsStartable does not imply that the target resource will actually be startable at a later time. This is because resource failures may prevent all of its relationships from being resolved at that later time.

See also "Location relationships" on page 73.

### Details on the principle behavior of the IsStartable relationship:

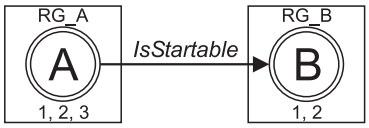
The IsStartable relationship causes the following behavior:
The IsStartable relationship defines that the source resource can only be placed on a node where the target resource is startable. This relationship is only considered if the source and target resources have nominal state online. When one of the resources (source or target) does not have a nominal state of online, the IsStartable relationship will be discarded together with the resources that have a nominal state of Offline.

The following example explains the behavior of the IsStartable relationship: Resource A and resource B are floating resources and members of the same resource group RG_A. Resource A can run on node1 and node2, and resource B on node2 and node3. An IsStartable relationship is defined from resource A to resource B.




Both members are started when the nominal state of the resource group is set to online. Based on the IsStartable relationship resource A and resource B are started on node2, as this node is the intersecting node for both resources. When resource B is in failed offline state on node 2, starting of the resource group RG_A does not start resource A, as no node exists where both resource A and resource B can be started.

The following example provides more information about the IsStartable relationship. In this scenario resource A can run on node1, node2, and node3, and is member of the resource group RG_A. Resource B can run on node1 and node2, and is member of resource group RG_B. An IsStartable relationship from resource A to resource B is defined.




The following describes the possible states in this example:

- RG_A's nominal state is set to online while RG_B is offline. Since the IsStartable relationship is only taken into consideration if the source and target resources have a nominal state of online (here RG_A and RG_B), and RG_B's nominal state

is offline in this case, the relationship will be ignored. Therefore resource A can either start on node1, or node2, or on node3.

- RG_A's nominal state is set to online while RG_B is already online. In this case the IsStartable relationship is taken into account and IBM Tivoli System Automation starts resource A on a node where resource B can start (node1 or node2).
- Due to a problem resource B cannot start on node1 and node2, and the nominal state of RG_B is online. Starting resource group RG_A causes that resource A cannot become online because resource B is not startable on the intersecting nodes node1 and node2.
- Due to a problem resource B cannot start on node1 and node2, and the nominal state of the resource group RG_B is offline. When resource group RG_A's nominal state is set to online, IBM Tivoli System Automation discards resource B, and the IsStartable relationship is ignored because of the desired state offline of resource group RG_B.

# Creating and administering relationships

## Creating a relationship

To create a relationship between a source resource and one or more target resources, you use the **mkrel** command.

The source resource must be a member of a resource group. The target resource does not have to be in a resource group.

For example, to define an AntiCollocated relationship for a source resource **FloatWebServerA** of class IBM.Application to target resource **FloatWebServerB** of class IBM.Application with condition 'IfOnline' and name 'Rel1', you would enter:

```
mkrel -p anticollocated -o ifonline -S IBM.Application:FloatWebServerA
      -G IBM.Application:FloatWebServerB Rel1
```

For further details, see either the **mkrel** man page, or the description of the mkrel command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Listing a relationship

To list a relationship, you use the **lsrel** command.

If you do not enter a relationship name, all relationships currently defined will be listed:

```
lsrel

Displaying Managed Relations :

Name Class:Resource:Node[Source]     ResourceGroup[Source]
Rel1 IBM.Application:FloatWebServerA RG_WebApp
```

If you specify a relationship name with the -M option, the persistent attributes of the specified relationship will be listed. For example, to list the attributes of the relationship Rel1 you would enter:

```
lsrel -M Rel1

Displaying Managed Relationship Information:
for Managed Relationship "Rel1".


Managed Relationship 1:
        Name                        = Rel1
        Class:Resource:Node[Source] = IBM.Application:FloatWebServerA
        Class:Resource:Node[Target] = {IBM.Application:FloatWebServerB}
        Relationship                = AntiCollocated
        Conditional                 = IfOnline
        ResourceGroup[Source]       = RG_WebApp
```

You may get a similar output if you list all relationships where IBM.Application:FloatWebServerA is the source of (-S option):

```
lsrel -S IBM.Application:FloatWebServerA

Displaying Managed Relationship Information:

Managed Relationship 1:
        Name                        = Rel1
        Class:Resource:Node[Source] = IBM.Application:FloatWebServerA
        Class:Resource:Node[Target] = {IBM.Application:FloatWebServerB}
```

```
Relationship                = AntiCollocated
Conditional                 = IfOnline
ResourceGroup[Source]       = RG_WebApp
```

For further details, see either the **lsrel** man page, or or the description of the lsrel command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Changing a relationship

To change a relationship, you use the **chrel** command.

For example, to change a relationship named Rel1 (created above) to AntiAffinity, you would enter:

```
chrel -p antiaffinity Rel1
```

For further details, see either the **chrel** man page, or the description of the chrel command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Removing a relationship

To remove a relationship between source and target resources, you use the **rmrel** command.

For example,to remove a relationship for a source resource **FloatWebServerA** of class IBM.Application, you would enter:

```
rmrel -S IBM.Application:FloatWebServerA
```

For further details, see either the **rmrel** man page, or the description of the rmrel command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

# Chapter 7. How IBM Tivoli System Automation processes the system information

This chapter describes in its first part the binding algorithm, also called binder. This is a IBM Tivoli System Automation internal function responsible for the node placements of all resources. The second part of this chapter discusses events which allow a resource group to become online. The third part of this chapter is about behavior patterns of IBM Tivoli System Automation .

## Location relationship resolution: Binding algorithm

The binder is invoked whenever a resource has to be started for which IBM Tivoli System Automation has not yet assigned a node placement. Resources which have a node location assigned are also called bound. An example is a floating resource A which could potentially run on several nodes. Here, the binding algorithm has to determine (bind) a node location for the floating resource considering all of its location relationships. Based on previous binding algorithm runs which already bound resources, a new solution has to be found. Binding solutions do not necessarily have to be non-ambiguous. Many constellations allow several alternative solutions where it is arbitrary which one is chosen by IBM Tivoli System Automation.

An example for an ambiguous scenario is a resource group with a collocated location relationship containing two floating resources A and B which can run on node1 and node2. When the group is started two alternative solutions are possible: either A and B are bound to node1 or both are bound to node2.


If the binding algorithm can find a solution for the node placement of all involved resources, the resource(s) are started. It is obvious that location relationships can lead to conflicting situations which have to be resolved.

For example, two floating resources A and B could be located on node1 and node2. Due to a performance constraint both resources must never run on the same node. You therefore must specify an AntiCollocated relationship from A to B and from B to A. It is assumed that resource A is already running on node 1. Then node 2 fails. If now a user started resource B, it would cause a location relationship conflict as resource A and B cannot be started on the same node. A perfect solution where both resources are running cannot be found in this situation. Therefore IBM Tivoli System Automation performs a specific conflict resolution called discarding step to resolve the situation.

It is possible that resources already online are part of the problem. These resources get an additional priority bonus of 10 to their priority set by the resource group.

The following section describes in detail IBM Tivoli System Automation's solution finding for location relationships and its conflict resolution handling. This whole process is called the binding algorithm

The binding algorithm consists of several steps:

1. **Discovery step: Determining configuration subsets for which the location relationships can be independently solved**

   The discovery algorithm consists of several substeps:

   a. **Step 1a: Find all involved resources (configuration subset)**

      Location relationships might separate a customer configuration into several configuration subsets which can be solved independently. The reason for this is that location relationships often affect only a subset of resources of the configuration. An example is the configuration with A --> Collocated --> B, B--> Collocated --> C, and D --> Collocated --> E. Here the location relationships for A, B, and C can be independently solved from D and E. For those two subsets all following steps are separately made .

   b. **Step 1b: Ignore all resources with OpState = Failed Offline**

      It is obvious that all instances which have an OpState of Failed Offline cannot contribute to a binding solution. Those instances are removed from the configuration subset which is used to find a binding solution. An example for this is a Resource Group R1 containing two floating resources A and B which can run on node1 and node2. The Group has a collocated parameter set which means that resource A and B have to be started on the same node. Assume that node2 is broken down which causes that the constituents of the floating resources A and B on node2 are in a Failed Offline state. Therefore those are removed from the configuration subset since instances on node2 will not help to solve the binding problem

   c. **Step 1c: Cleaning up resource groups which cannot be started.**

      In case mandatory resource group members are in a Failed Offline state, the resource group cannot be started according to the resource group behavior. Therefore all other resource group members of such a resource group have to be stopped.

      An example is resource group R1 with floating resource A and B as described above. If floating resource A cannot be started on either of the nodes due to an application error, and if it is a mandatory resource group member, the floating resource B is also stopped (see resource group members).

2. **Perfect solution step: Try to find a 'perfect' solution**

   At first, the recovery resource manager tries to find a perfect solution of all involved location relationships for a configuration subset. In this step it tries to find bindings as described in "Location relationships" on page 73. Since in this first step the goal is to find a perfect solution, all Affinity and AntiAffinity relationships are treated as if they were pure Collocated and AntiCollocated relationships. In addition, even resources which are Offline and are not intended to start are also tried to be bound if necessary. If no location relationship conflict occurs, the necessary resources are bound and the binding algorithm is done. As a next step IBM Tivoli System Automation can start those resources which have to be started.

   There are situations in which this binding step gets into a conflict situation with contradicting constraints that cannot be overcome. To resolve this IBM Tivoli System Automation provides a discarding step consisting of several substeps as described below.

3. **Discarding step: Resolve situations with conflicting location relationships**

   The discarding step consist of a a number of substeps:

   a. **Step 3a: Ignore all Affinity and AntiAffinity relationships**

The first approach to overcome the conflicting situation is to ignore all Affinity and AntiAffinity relationships since those are 'soft' location relationships. Based on the previous bindings IBM Tivoli System Automation tries to find a solution for the resources which have to be bound. Since all Affinity and AntiAffinity relationships are ignored, the location relationships are simplified and the probability that a binding solution can be found is increased. In case a solution can be found, the sacrifice step is left. But there is still the chance that the conflicting situation cannot be overcome. Then the next level of the sacrifice step is reached.

b. **Step 3b: Ignore all resources with OpState = Offline and which do not have to be started**

If ignoring all Affinity and AntiAffinity relationships did not help to find a solution for the binding problem (see step 3a on page 88) then the next level is to ignore all resources from the binding evaluation which are Offline and which are currently not intended to be started. This increases the possibility that a binding solution can be found.

In case there is a binding solution available, then the sacrifice step is left. Otherwise the next step of the discarding process is reached.

An example is ResourceGroup R1 containing floating resource A, and resource group R2 containing floating resource B, and a relationship A AntiCollocated B. Floating resource A and B can run on node1 and node2, but node2 is broken down.


Now R1's nominal state is set to online which causes that resource A has to be bound before it can be started. At first IBM Tivoli System Automation tries to find a perfect solution. Therefore it tries to bind A and B. But here a solution cannot be found. Then IBM Tivoli System Automation ignores all Affinity and AntiAffinity relationships, which does not provide a solution either. Then it ignores all resources with an Offline state and which do not have to be started. This causes that resoure B is ignored for the evaluation. Now it is possible to bind resource A to node1.

The non-mandatory resources are removed from the binding set in the following order:

1) Resources that are not startable are removed first.
2) If no resource could be removed, the resources that are currently offline are considered next.
3) If still no resource could be removed, all non-mandatory resources are removed.

c. **Step 3c: Stopping least important resource group members**

The next level of the sacrifice step is to stop resource group members and ignore those members in the binding evaluation. Since each resource group has a priority value assigned, resource group members of the group(s) with the lowest priority are stopped first, and then a binding solution is tried to find without them. In case this does not satisfy the binding constraints, resource group members with the next group priority level are chosen. In addition to the priority schema, the stopping and removing of resource group members is performed in two substeps: First only all non-mandatory members of group priority level are stopped and ignored for the binding solution. Only if this does not help to solve the conflicting situation, then also the mandatory members of the same group priority level are stopped

and removed from the binding evaluation. If the conflict still exists, the next lowest group priority is taken and all group members are stopped as just described. This is done iteratively until a binding solution can be found.

**Hints**:

- Outer groups must have the same or a higher priority than the inner groups. Otherwise the outer groups would be discarded before the inner groups. But if the outer groups are discarded, the inner groups are stopped automatically.
- Resource groups that are non-mandatory members of another resource group should have a lower priority than resource groups that are mandatory members of the same resource group. Otherwise mandatory members may be discarded.

## Events that might allow a resource group to become Online

All "root" resource groups whose NominalState attribute is Online will be *automated*: this means, an attempt will be made to start such "root" resource groups and the managed resources within these groups, *providing* the managed relationships of the managed resources within the resource groups can be satisfied.

If a resource group or a managed resource cannot be brought Online (when one or more of its member resources fail completely to reach the Online required state), the resource group is in an Offline state. The resource group remains Offline until an event occurs that informs IBM Tivoli System Automation that it should again attempt to start the resource group.

These are the possible events which might cause an Offline resource group to become Online:

- Changing the *AllowedNodes attribute* (explained in "AllowedNode attribute" on page 35) of the resource group, for example to include an additional node where the resource group can be started. For details, see the description of the chrg command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.
- Removing a managed resource from a resource group. As a result, the other member resources might then be startable because a resource which cannot be started is removed. For details, see the description of the rmrgmbr command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.
- Adding a resource to a resource group which is a target resource of a managed relationship, but which is currently not a member of a resource group. It will be automated by IBM Tivoli System Automation then. As a result, the other member resources might then be startable because a managed relationship can be satisfied. For details, see the description of the addrgmbr command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.
- Starting a resource which is not controllable by IBM Tivoli System Automation and which has a managed relationship to a resource group member. As a result, the resource goes Online and the managed relationship is satisfied. This might allow the resource group to also be brought online. To start the resource, you might use the RMC **startrsrc** command (for details, see the man page for this command).
- Adding a constituent to an aggregate that will make the aggregate resource available on more nodes, and may result in IBM Tivoli System Automation being able to satisfy all of the managed-relationship constraints. If the constituent is a piece of hardware, this would require that you install the hardware, or define it

correctly. If the resource is a floating resource, you add a constituent by adding a node name to the NodeNameList attribute. For details, refer to the RMC documentation and man pages.

- A new resource is found by an equivalency that uses a dynamic select string. As a result, this resource is added to the equivalency, and may resolve a managed relationship to this equivalency.
- Making a managed resource NotMandatory, which allows this resource to be sacrificed. As a result, the other managed resources can be started. For details, see the description of the chrgmbr command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.
- Performing a Reset on an aggregate or one of its constituents after a failure has been fixed. As a result, the resource will be Offline, and can then be started by IBM Tivoli System Automation. For details, see the man page for the RMC **resetrsrc** command.
- A node that was Offline becomes Online. As a result, IBM Tivoli System Automation may be able to bring the resource group online.
- Changing the priority attribute of a resource group (explained in "Priority attribute" on page 37). A resource group might not be startable because of a priority conflict with another resource group. In this case, increase the priority of the group you want to start or decrease the priority of the other group. The changes will take effect when you next start the group.
- Stopping a higher-priority resource group, which prevents a lower-priority resource group from starting. As a result, a managed relationship conflict is avoided. For details, see the description of the chrg command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

If a resource group is currently at its NominalState value, the following events may cause additional automation actions:

1. The NominalState attribute value changes from Offline to Online.
2. The NominalState attribute value changes from Online to Offline.

# Behavior Patterns of IBM Tivoli System Automation for Multiplatforms

This section describes how IBM Tivoli System Automation behaves and reacts in certain situations.

## General considerations

The following describes issues relating to the StartCommand, MonitorCommand, and StopCommand.

### StartCommand issues

IBM Tivoli System Automation for Multiplatforms uses the command specified in the StartCommand attribute of a resource to bring a resource Online. The StartCommand of a resource is executed in the following situations:

- Immediately, after the NominalState attribute of a resource group has been changed to Online, and all start dependencies of this resource are satisfied.
- Immediately, after the OpState of a resource has changed from Online to Offline caused by a failure of the resource (Note: This is not true, if the NominalState of the resource group has been changed to Offline, or if the resource has been stopped/forced down by IBM Tivoli System Automation to satisfy a dependency to another resource.)
- When the StartCommand had already been executed for a resource but the resource is still Offline at the time the Online timeout is reached **and** the

RetryCount for the number of StartCommand executions has not been reached, the Online timeout (in seconds) for a resource is calculated using the following formula:

```
MAX( StartCommandTimeout, MonitorCommandPeriod, MonitorCommandTimeout ) + 10
```

Note that +10 is not an absolute value, as IBM Tivoli System Automation does not use a real timer. The IBM Tivoli System Automation daemons are woken up frequently, and this may result in the additional value, which may be in the range of 10 to 13 seconds. Also note also that this Online time-out is only evaluated in case the resource did not change its OpState during the previous execution of the StartCommand (e.g. to Pending Online or Online). Then the Online time-out timer is canceled, and the StartCommand of the resource is executed immediately after the OpState of the resource has changed to Offline again.

By default, the StartCommand is executed synchronously by IBM Tivoli System Automation, meaning that IBM Tivoli System Automation waits until the command completes and it obtains knowledge of any return code.

Furthermore there is an attribute StartCommandTimeout for each resource which determines how long it takes at maximum to execute the StartCommand. If the StartCommand does not return within the StartCommandTimeout time period, the StartCommand is killed by IBM Tivoli System Automation using the SIGKILL command. If this happens, a message is logged into the system log of that node. However this may lead to problems if an application process that is started within the StartCommand does not return control IBM Tivoli System Automation. In this case the application process is killed every time after the StartCommandTimeout is reached, which is why IBM Tivoli System Automation cannot start this application as a resource. To get this working the application process must be detached from the calling StartCommand by using one of the following methods:

- Redirect all file handles to a file and start the application process in the background, e. g.

  ```
  /usr/bin/application  >/outputfile  2>&1   &
  ```

- Create a little wrapper application that uses the 'setsid()' C-function to get the application process detached from the calling StartCommand.
- If the above methods do not work or are not appropriate for a certain application, then the value of the attribute RunCommandsSync of the resource has to be set to 0. In this case IBM Tivoli System Automation does not honor the StartCommandTimeout attribute for this resource, and therefore the StartCommand and all its child processes may stay until forever on this node. But in this case IBM Tivoli System Automation does not wait for any return code of this StartCommand and therefore a resource is not failed over even if the StartCommand failed. Instead, the StartCommand is executed again if the resource does not come Online during the Online time-out period until the RetryCount is reached.

## MonitorCommand issues

The MonitorCommand of an IBM.Application resource is used by IBM Tivoli System Automation to determine the OpState of this resource on a node. IBM Tivoli System Automation starts monitoring a resource at the time when it is added to a resource group or equivalency. The resource is monitored on any node on which it is allowed to run (NodeNameList).

After the first execution, the MonitorCommand is executed in the frequency defined in the MonitorCommandPeriod attribute. This monitoring of the resource

now goes on forever on every node on which the resource is defined, until the resource is removed from the resource group or equivalency.

Starting with IBM Tivoli System Automation release 1.2 the MonitorCommand is also executed immediately after the StartCommand or StopCommand of a resource has finished execution (only for synchronous commands, if RunCommandsSync attribute for this resource is set to 1, which is the default). This has been introduced to enhance the performance of the start/stop of an entire resource group, as now the OpState of the resource is immediately checked after the StartCommand or StopCommand has finished. After this execution of the MonitorCommand, the frequency of MonitorCommandPeriod seconds is honored again, meaning that the next MonitorCommand is executed after MonitorCommandPeriod seconds.

Starting with IBM Tivoli System Automation release 2.2, the value of the attribute MonitorCommandPeriod can be lower than that of the attribute MonitorCommandTimeout for a given resource. This is possible because the MonitorCommandPeriod actually is the waiting period between the end of the last execution of the MonitorCommand and the start of the next. As a result, the MonitorCommandTimeout can now be specified independently from the MonitorCommandPeriod, and thus a frequent MonitorCommand execution can be realized even with long running MonitorCommand scripts. However, it is still recommended that the MonitorCommand be kept as efficient as possible.

There are two issues regarding the MonitorCommand that should be kept in mind to avoid trouble or strange behavior:

1. The MonitorCommand is executed on all nodes the resource is allowed to run on (that are defined in the resources NodeNameList attribute). If a resource should be down (NominalState of the resource group is Offline) and an operator starts this resource manually, IBM Tivoli System Automation will notice this with the MonitorCommand of that resource and finally execute the StopCommand for this resource to bring it back Offline again. This is how IBM Tivoli System Automation is designed: automate resources. If it is necessary to bring a single resource of a resource group Online (or Offline), for instance to perform a backup, then an IBM Tivoli System Automation request has to be used (rgreq command). This will overrule the NominalState of the resource group and will allow a resource of a resource group to be started, even if the NominalState of the resource is Offline.
2. There is an attribute MonitorCommandTimeout, which will result in a SIGKILL command against a running MonitorCommand if this has not finished before the time out has been reached. If the MonitorCommand has been killed, a message will be logged into the system log of that node, and the OpState of the resource will be set to Unknown. In this case, IBM Tivoli System Automation will not go on automating this or any dependent resource until a meaningful state can be determined again. If this message is seen frequently in the system log, the value of the MonitorCommandTimeout attribute should be checked and adjusted if necessary.

## StopCommand issues

By default, the StopCommand is executed synchronously by IBM Tivoli System Automation, meaning that IBM Tivoli System Automation waits for the command to finish and obtain knowledge of any return code. Furthermore there is an attribute StopCommandTimeout for each resource which determines how long it takes at most to execute the StopCommand. If the StopCommand does not return within the StopCommandTimeout time period, the StopCommand is killed by IBM

Tivoli System Automation using the SIGKILL command. If this happens a message is logged into the system log of that node.

If the StopCommand for a resource has been executed but the OpState of the resource does not change to Offline within the Offline timeout period, IBM Tivoli System Automation issues a reset operation against the resource. The Offline timeout period (in seconds) for a resource is calculated using the following formula:

```
MAX( StopCommandTimeout, MonitorCommandPeriod, MonitorCommandTimeout ) + 10
```

Note that +10 is not an absolute value as IBM Tivoli System Automation does not use a real timer. The IBM Tivoli System Automation daemons are woken up frequently, and this may result in the additional value, which may be in the range of 10 to 13 seconds.

Issuing a reset operation against a resource results in a second execution of the StopCommand, but this time a special environment variable (SA_RESET) is set to 1. This can be exploited within the StopCommand of a resource to achieve an enhanced force behavior, for example:

```
#/bin/sh
# A sample stop/reset automation script for the lpd application
if [ $SA_RESET == 1 ]; then
  killall -9 lpd
  exit $?
else
  /etc/init.d/lpd stop
  exit $?
fi
```

If a second execution of the StopCommand is not desired, the script could just exit, for example:

```
#/bin/sh
# A sample stop/reset automation script for the lpd application
if [ $SA_RESET == 1 ]; then
  exit $?
else
  /etc/init.d/lpd stop
  exit $?
fi
```

If the resource still does not stop after the second execution of the StopCommand, the OpState of the resource is set to Stuck Online, which means that manual intervention is now required to stop the resource. IBM Tivoli System Automation will resume automating the resource only after the problem has been resolved by an operator.

This behavior is introduced in IBM Tivoli System Automation for Multiplatforms, release 2.2. However, rewriting existing policy scripts is not required as they will work as before. The only difference is that the StopCommand is now executed twice for any given resource if its operational state does not change to Offline within the Offline timeout period.

## How IBM Tivoli System Automation reacts to the possible OpState changes of a resource that is online on a node

The following sample configuration is used for the discussion in the next section:

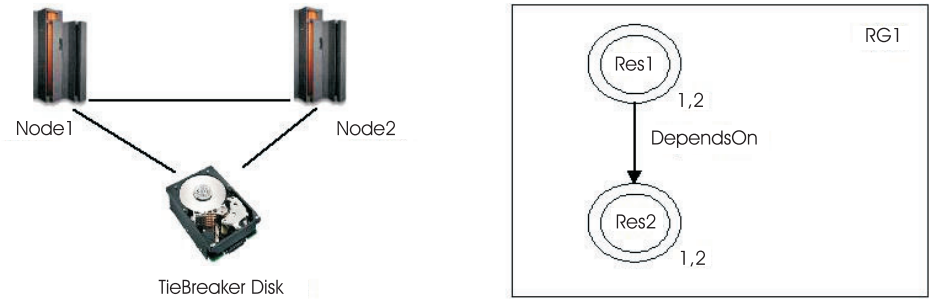



*Figure 3. Sample configuration setup*

The setup of this sample configuration is as follows:

- Cluster of 2 nodes.
- Disk TieBreaker
  - Node1: production system.
  - Node2: stand-by system.
- Resource group: **RG1.** with
  - Floating resource: Res1
  - Floating resource: Res2
  - Relationship: Res1 **DependsOn.** Res2
- Resources are **Online** on Node1.

Figure 4 on page 96 shows the typical OpState transitions for resources that are automated by IBM Tivoli System Automation:

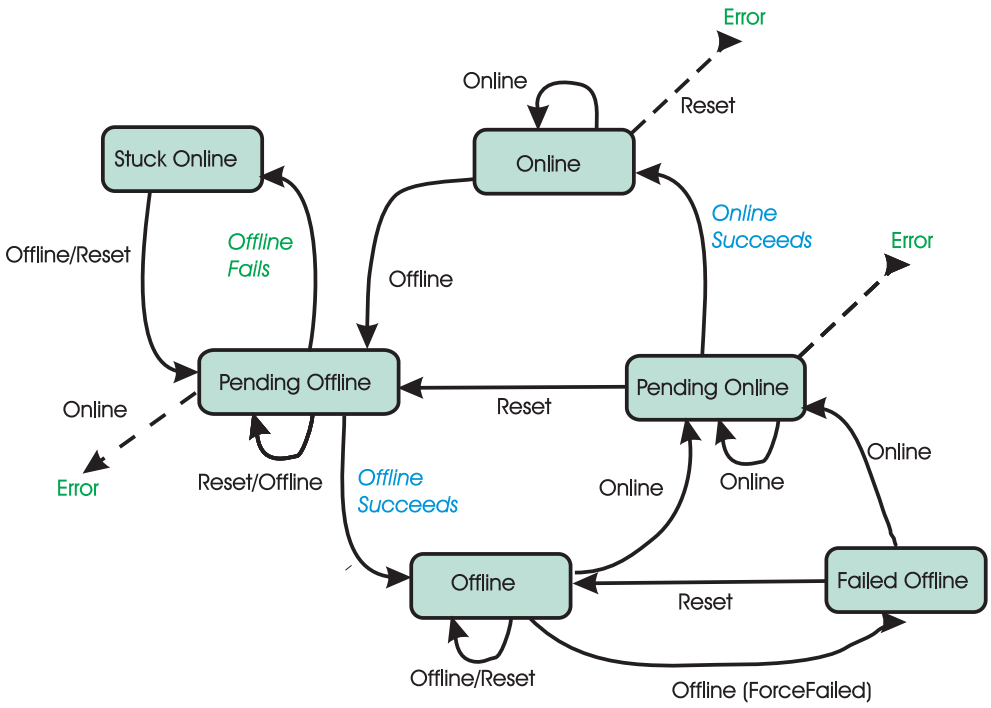



*Figure 4. Operational state transitions of automated resources*

There are seven values the OpState of a resource can have. The OpState of a resource is determined by IBM Tivoli System Automation with the MonitorCommand, the actual OpState of a resource is provided to IBM Tivoli System Automation with the return code of the MonitorCommand. Note that it is sufficient for a MonitorCommand to return the OpState values Online and Offline to IBM Tivoli System Automation, the other OpState values a resource can have may be exploited optionally.

Some OpState values like Unknown or Failed Offline may be also set by IBM Tivoli System Automation, for instance, the OpState Unknown is set for a resource if the MonitorCommand for this resource timed out. IBM Tivoli System Automation therefore has no knowledge about the OpState of this resource anymore.

The following two tables illustrate how IBM Tivoli System Automation reacts to an OpState change of the resources Res1 and Res2 from the example above. Note that the tables in this chapter contain all possible OpState values of a resource, even if a particular OpState does not make sense in this situation. The columns which contain these unlikely OpState values are preceeded by the word 'unlikely' in the following tables.

## OpState change of resource Res1

The current status of Res1 and Res2 on node1 is Online. The following table shows the actions that IBM Tivoli System Automation performs depending on the return value of the MonitorCommand for Res1.

*Table 5. System Automation actions regarding OpState changes of resource Res1*

| MonitorCommand (OpState) | | First action of System Automation | Second action of System Automation |
|---|---|---|---|

*Table 5. System Automation actions regarding OpState changes of resource Res1 (continued)*

| | | | |
|---|---|---|---|
| RC=0 (Unknown) | => | Nothing, wait for next MonitorCommand with RC<>0 | Nothing |
| RC=1 (Online) | => | Nothing | Nothing |
| RC=2 (Offline) | => | Start Res1 | Nothing |
| RC=3 (Failed Offline | => | Stop Res2 | After Res2 is Offline, start both resources on node2 in correct order |
| RC=4 (Stuck Online) | => | Nothing: wait for Operator action | Nothing |
| RC=5 (Pending Online | => | Unlikely, wait for Online | Nothing |
| RC=6 (Pending Offline) | => | Wait for Offline | Nothing |

### OpState change of resource Res2

The current status of Res1 and Res2 on node1 is Online. The following table shows the actions that IBM Tivoli System Automation performs depending on the return value of the MonitorCommand for Res2.

*Table 6. System Automation actions regarding OpState changes of resource Res2*

| MonitorCommand (OpState) | | First action of System Automation | Second action of System Automation |
|---|---|---|---|
| RC=0 (Unknown) | => | Nothing, wait for next MonitorCommand with RC<>0 | Nothing |
| RC=1 (Online) | => | Nothing | Nothing |
| RC=2 (Offline) | => | Force down Res1 | After Res1 is Offline, start Res2, after Online of Res2 -> start Res1 |
| RC=3 (Failed Offline | => | Force down Res1 | Start both resources on node2 in correct order |
| RC=4 (Stuck Online) | => | Nothing: wait for Operator action | Nothing |
| RC=5 (Pending Online | => | Unlikely, wait for Online | Nothing |
| RC=6 (Pending Offline) | => | Wait for Offline | Nothing |

## How System Automation composes the OpState of a resource group

IBM Tivoli System Automation for Multiplatforms is a policy-based automation product. The control point for the automation is the resource group level, meaning that an Operator usually starts or stops an entire resource group rather than starting or stopping single resources. This is done by changing the NominalState attribute of a resource group to Online or Offline. Immediately after changing this attribute IBM Tivoli System Automation will decide which resources needs to be started or stopped to meet the rules of the changed policy.

The OpState (Operational State) attribute of a resource group is an aggregation of the OpState attributes of all resources contained in that resource group **in relation to the NominalState value of the resource group**. So if the NominalState of a resource group has been changed to Online, the OpState of this resource group is showing Pending Online until all of the resources in that resource group are Online. Finally, if all resources of that resource group have reached the value of the NominalState attribute of the resource group, the OpState of the resource group changes to Online, and this value of the OpState attribute of a resource group can now be used to monitor the status of the resources in that group.

**Note:** Member resources may have a different OpState than the containing group when requests were issued directly against the member resources. For more information, see "Using requests to start and stop resource groups and resources" on page 170.

The following table shows how IBM Tivoli System Automation composes the value of the OpState attribute of a resource group based on the OpState of the two contained resources Res1 and Res2 from the above example. Note that this picture becomes more complex the more resources are contained in a single resource group.

*Table 7. Resource group OpState determination*

| OpState of Res2 | OpState of Res1 | | OpState of Resource Group | Action of System Automation |
|---|---|---|---|---|
| Unknown | Unknown | => | Unknown | Nothing |
| Offline | Offline | => | Offline | Nothing |
| Pending Online | Offline | => | Pending Online | Wait until Res2 is Online |
| Online | Offline | => | Pending Online | Start Res1 |
| Online | Pending Online | => | Pending Online | Wait until Res1 is Online |
| Online | Online | => | Online | Nothing |
| Online | Pending Offline | => | Pending Offline | Wait until Res1 is Offline |
| Online | Failed Offline | => | Pending Offline | Stop Res2 |
| Pending Offline | Offline | => | Pending Offline | Wait until Res2 is Offline |
| Failed Offline | Offline | => | Offline | Nothing |

## How System Automation reacts to OpState changes of a resource that is started or stopped

IBM Tivoli System Automation usually automates resources based on the DesiredState of the resource group and the OpState value of the resource. The goal is to achieve and maintain a state in which the OpState and the DesiredState of the resource are the same. Furthermore, IBM Tivoli System Automation takes action if the OpState of a resource is changing, for example, when a resource that was running is now monitored Offline.

Another trigger for automation actions is the return code of the StartCommand. If the StartCommand returns an error (a non-zero return code) and the resource is not monitored Online, then IBM Tivoli System Automation also takes action and

performs a fail over of the resource to another eligible node. The following sections describe the actions that are performed by IBM Tivoli System Automation if the OpState of a resource changes during or shortly after the execution of the StartCommand or StopCommand of that resource.

The enhancements introduced in IBM Tivoli System Automation V2R2 enforce a more compliant implementation of the MonitorCommand of a resource. This is especially important at the time when the StartCommand for a resource is executed. At that time, IBM Tivoli System Automation knows that the OpState of the resource is Pending Online, and so this OpState is set for the resource. The only exceptions are:

- the OpState reported by the MonitorCommand changes to Online (in which case the automation goal is reached)
- the OpState reported by the MonitorCommand changes to Failed Offline (which informs IBM Tivoli System Automation that the resource is broken and cannot be started on that node without prior manual intervention)

In a similar way, this applies to the execution of the StopCommand, where the OpState of the resource is set to Pending Offline during that period. Possible exceptions are:

- the MonitorCommand returns Offline or Failed Offline (which indicate that the automation goal has already been reached)
- the MonitorCommand returns Stuck Online (which indicates that manual intervention is required because the resource cannot be stopped by IBM Tivoli System Automation)

## StartCommand

The following tables illustrate how IBM Tivoli System Automation reacts to OpState changes during the execution of the StartCommand. There is one table for each of the three possible situations where the MonitorCommand can report an OpState change:

1. The StartCommand is still under execution (long running StartCommand).
2. The StartCommand has successfully finished (this is the normal situation).
3. The StartCommand has finished with an error or has timed out.

**StartCommand is still under execution:**

*Table 8. System Automation actions and StartCommand still under execution*

| StartCommand | MonitorCommand | Action of System Automation |
|---|---|---|
| StartCommand started, but not finished. | RC=0 (Unknown) | OpState is set to Pending Online, Wait for Online |
| | RC=1 (Online) | Start other resources, if any |
| | RC=2 (Offline) | OpState is set to Pending Online, Wait for Online |
| | RC=3 (Failed Offline) | Stop command against the resource and then fail over to another node, probably force other dependent resources down |
| | RC=4 (Stuck Online) | Unlikely<br>OpState is set to Pending Online, Wait for Online |
| | RC=5 (Pending Online) | OpState is set to Pending Online, Wait for Online |
| | RC=6 (Pending Offline) | Unlikely<br>OpState is set to Pending Online, Wait for Online |

Note that once the MonitorCommand has reported the resource as Online, IBM Tivoli System Automation does not take care about the still running StartCommand anymore, as the goal to bring the resource Online is already achieved.

If the OpState for the resource is reported as Failed Offline, the resource is stopped, and a failover is initiated. All other potential OpState values are mapped to Pending Online, and System Automation waits for the OpState to change to Online.

**StartCommand successfully finished:** This table describes the typical behavior of IBM Tivoli System Automation:

*Table 9. System Automation actions after StartCommand successfully finished*

| StartCommand | MonitorCommand | Action of System Automation |
|---|---|---|
| RC=0 (successful) and actual retry count < RetryCount (samctrl) | RC=0 (Unknown) | OpState is set to Pending Online, Wait for Online |
| | RC=1 (Online) | Start other resources, if any |
| | RC=2 (Offline) | OpState is set to Pending Online, Wait for Online<br>After Online timeout: perform start retry, increase retry count |
| | RC=3 (Failed Offline) | Stop command against the resource and then fail over to another node, probably force other dependent resources down |
| | RC=4 (Stuck Online) | Unlikely<br>OpState is set to Pending Online, Wait for Online |
| | RC=5 (Pending Online) | OpState is set to Pending Online, Wait for Online |
| | RC=6 (Pending Offline) | Unlikely<br>OpState is set to Pending Online, Wait for Online |
| RC=0 (successful) and actual retry count = RetryCount (samctrl) and after Online timeout | | Set resource to Failed Offline, send stop command against the resource and then fail over to another node. probably force down other dependent resources |

**StartCommand finished with an error or timed out:** The following table describes the behavior of IBM Tivoli System Automation if the StartCommand for a resource returns with an error or times out, depending on the OpState of the resource:

*Table 10. System Automation actions after StartCommand finished with an error or timed out*

| MonitorCommand | StartCommand | Action of System Automation |
|---|---|---|
| RC=0 (Unknown) | RC=1 (not zero) not successful, or timed out | After MonitorCommand reported Unknown, IBM Tivoli System Automation waits for MonitorCommand returns RC<>0, especially the **StartCommand RC (or time out) is ignored**.<br><br>If next valid Monitor is Online, resource stays Online, if next valid Monitor is Offline, retry of StartCommand is performed. |

*Table 10. System Automation actions after StartCommand finished with an error or timed out (continued)*

| MonitorCommand | StartCommand | Action of System Automation |
|---|---|---|
| RC=1 (Online) | RC=1 (not zero) not successful, or timed out | After MonitorCommand had returned Online, **StartCommand RC (or time out) is ignored**. No further actions, resource stays Online. |
| RC=2 (Offline) | RC=1 (not zero) not successful, or timed out | Immediately after the StartCommand returns with RC=1, the resource is stopped by IBM Tivoli System Automation, and then a fail over takes place. |
| RC=3 (Failed Offline) | RC=1 (not zero) not successful, or timed out | If a MonitorCommand did return 3 (Failed Offline) previously, then a failover already takes place, and the unsuccessful execution of the StartCommand has no additional effect |
| RC=4 (Stuck Online) | RC=1 (not zero) not successful, or timed out | Unlikely, immediately after the StartCommand returns with RC=1, the resource is stopped by IBM Tivoli System Automation, and then wait for Operator action. |
| RC=5 (Pending Online) | RC=1 (not zero) not successful, or timed out | Immediately after the StartCommand returns with RC=1, the resource is stopped by IBM Tivoli System Automation, and then a fail over takes place after the resource is reported Offline. |
| RC=6 (Pending Offline) | RC=1 (not zero) not successful, or timed out | Unlikely, immediately after the StartCommand returns with RC=1, the resource is stopped by IBM Tivoli System Automation, and then a fail over takes place after the resource is reported Offline. |

Note that the return code of the StartCommand in the table above is ignored if the MonitorCommand already monitored this resource as Online. In this case the result of the two commands is inconsistent: the StartCommand tells IBM Tivoli System Automation that the start of the resource has failed, but the MonitorCommand already has monitored the resource Online. This is considered as a script error - either within the StartCommand or within the MonitorCommand.

Note also that the return code of the StartCommand has no effect if the resource is monitored as Unknown. In this case IBM Tivoli System Automation waits for a

valid (non-Unknown) OpState of the resource and the automation will proceed after it receives the next valid (non-Unknown) return code of the MonitorCommand.

## StopCommand

The following tables illustrate how IBM Tivoli System Automation reacts to OpState changes during the execution of the StopCommand. There is one table for each of the three possible situations where the MonitorCommand can report an OpState change:

1. The StopCommand is still under execution (long running StopCommand).
2. The StopCommand has successfully finished (this is the normal situation).
3. The StopCommand has finished with an error or has timed out.

**StopCommand is still under execution:**

*Table 11. System Automation actions and StopCommand still under execution*

| StopCommand | MonitorCommand | Action of System Automation |
|---|---|---|
| StopCommand started, but not finished. | RC=0 (Unknown) | OpState is set to Pending Offline, Wait for Offline |
| | RC=1 (Online) | OpState is set to Pending Offline, Wait for Offline |
| | RC=2 (Offline) | Go on stopping other resources |
| | RC=3 (Failed Offline) | Go on stopping other resources |
| | RC=4 (Stuck Online) | Wait for Operator action |
| | RC=5 (Pending Online) | Unlikely<br>OpState is set to Pending Offline, Wait for Offline |
| | RC=6 (Pending Offline) | OpState is set to Pending Offline, Wait for Offline |

Note that once the MonitorCommand has reported the resource as Offline or Failed Offline, IBM Tivoli System Automation does not take care about the still running StopCommand anymore, as the goal to bring the resource Online is already achieved.

If the OpState for the resource is reported as Stuck Online, IBM Tivoli System Automation waits for the intervention by an operator, because the resource cannot be stopped. All other potential OpState values are mapped to Pending Offline, and IBM Tivoli System Automation waits for the OpState to change to Offline.

**StopCommand successfully finished:** This table describes the typical behavior of IBM Tivoli System Automation:

*Table 12. System Automation actions after StopCommand successfully finished*

| StopCommand | MonitorCommand | Action of System Automation |
|---|---|---|
| RC=0 (successful)<br>and<br>( (Offline time out is not reached)<br>or<br>(reset sent against the resource and reset timeout is not reached) ) | RC=0 (Unknown) | OpState is set to Pending Offline, Wait for Offline |
| | RC=1 (Online) | OpState is set to Pending Offline, Wait for Offline |
| | RC=2 (Offline) | Go on stopping other resources |
| | RC=3 (Failed Offline) | Go on stopping other resources |
| | RC=4 (Stuck Online) | Wait for Operator action |
| | RC=5 (Pending Online) | Unlikely<br>,OpState is set to Pending Offline, Wait for Offline |
| | RC=6 (Pending Offline) | OpState is set to Pending Offline, Wait for Offline |
| RC=0<br>(successful)<br>and<br>after Offline timeout | | OpState is set to Pending Offline, Reset sent against the resource |
| RC=0<br>(unsuccessful)<br>and<br>after reset timeout | | OpState is set to Stuck Online, Wait for Operator action |

**StopCommand finished with an error or timed out:**

*Table 13. System Automation actions after StopCommand finished with an error or timed out*

| StopCommand | MonitorCommand | Action of System Automation |
|---|---|---|
| RC=1<br>(not zero; not successful, or timed out)<br>and<br>( (Offline time out is not reached)<br>or<br>(reset sent against the resource and reset timeout is not reached) ) | RC=0 (Unknown) | OpState is set to Pending Offline, Wait for Offline |
| | RC=1 (Online) | OpState is set to Pending Offline, Wait for Offline |
| | RC=2 (Offline) | Go on stopping other resources |
| | RC=3 (Failed Offline) | Go on stopping other resources |
| | RC=4 (Stuck Online) | Wait for Operator action |
| | RC=5 (Pending Online) | Unlikely<br>OpState is set to Pending Offline, Wait for Offline |
| | RC=6 (Pending Offline) | OpState is set to Pending Offline, Wait for Offline |
| RC=1<br>(unsuccessful)<br>and<br>after Offline timeout | | OpState is set to Pending Offline, Reset sent against the resource |

*Table 13. System Automation actions after StopCommand finished with an error or timed out (continued)*

| StopCommand | MonitorCommand | Action of System Automation |
|---|---|---|
| RC=1<br>(unsuccessful)<br>and<br>after reset timeout | | OpState is set to Stuck Online, Wait for Operator action |

IBM Tivoli System Automation does not honor the return code of the StopCommand. The OpState of the resource is set to Pending Offline until its OpState changes to Offline, Failed Offline, or Stuck Online. If the OpState does not change to Offline or Failed Offline within the Offline timeout period, IBM Tivoli System Automation issues a reset operation against the resource, which triggers an execution of the StopCommand. If the OpState still does not change to Offline, Failed Offline, or Stuck Online within the reset timeout period, the resource is finally set to Stuck Online to indicate that an operator must take action to stop the resource.

## How System Automation reacts if a resource is Online on a certain node and the MonitorCommand reports an OpState for the resource on another node at the same time

The following table shows the actions IBM Tivoli System Automation will perform, if a resource is Online on a node, and the MonitorCommand returns a certain OpState for this resource on another node.

*Table 14. System Automation actions after the MonitorCommand reports an OpState change for the resource on another node*

| MonitorCommand on stand-by Node | Action of System Automation |
|---|---|
| RC=0 (Unknown) | No automation action possible for this resource until MonitorCommand returns with RC<>0. |
| RC=1 (Online) | The resources on both nodes are stopped (and all resources, that depend on that resource). Then the resources are started on one of the nodes again. |
| RC=2 (Offline) | No action, this is usual OpState of the resource on a stand-by node. |
| RC=3 (Failed Offline) | No action, but no fail over is possible to this node anymore. |
| RC=4 (Stuck Online) | Unlikely, script error (requires Online OpState before ...) |
| RC=5 (Pending Online) | Same as Online |
| RC=6 (Pending Offline) | Unlikely, script error (requires Online OpState before ...) |

Most important to notice is the fact, that IBM Tivoli System Automation will stop the resource on both nodes, if it is monitored Online on more than one node at a time, and not just the resource on the stand-by node. This also implies all resources that have a dependency to this resource may also be stopped. For instance all resources that have a DependsOn relationship to this resource will be stopped too.

It is therefore recommended not to start and stop applications and resources manually that are under control of IBM Tivoli System Automation.

# Chapter 8. Using the operations console

This section gives an overview of the operations console and shows how to use it in direct access mode, which is the mode available for users of the base component of IBM Tivoli System Automation.

For users of the end-to-end automation management component of IBM Tivoli System Automation for Multiplatforms, two more modes are available. These are described in the *IBM Tivoli System Automation for Multiplatforms End-to-End Automation Management Component Administrator's and User's Guide*.

## Overview - what is the operations console?

The operations console is a browser-based graphical user interface that runs in IBM Integrated Solutions Console (ISC). You use the operations console to monitor and manage resources managed by IBM Tivoli System Automation. It consists of the following parts:

- A WebSphere Application Server embedded in Integrated Solutions Console.
- Integrated Solutions Console, which runs as an application in the WebSphere Application Server. Integrated Solutions Console can host multiple application front-ends. The operation console of IBM Tivoli System Automation for Multiplatforms is one of these applications.
- The operations console, which is the actual front-end that is used by the operators, runs within IBM Integrated Solutions Console.
- Operators use a Web browser to contact IBM Integrated Solutions Console and display the operations console.

The following figure shows the setup of the operations console both for the user of the base component and for the user of the end-to-end automation management component of IBM Tivoli System Automation.

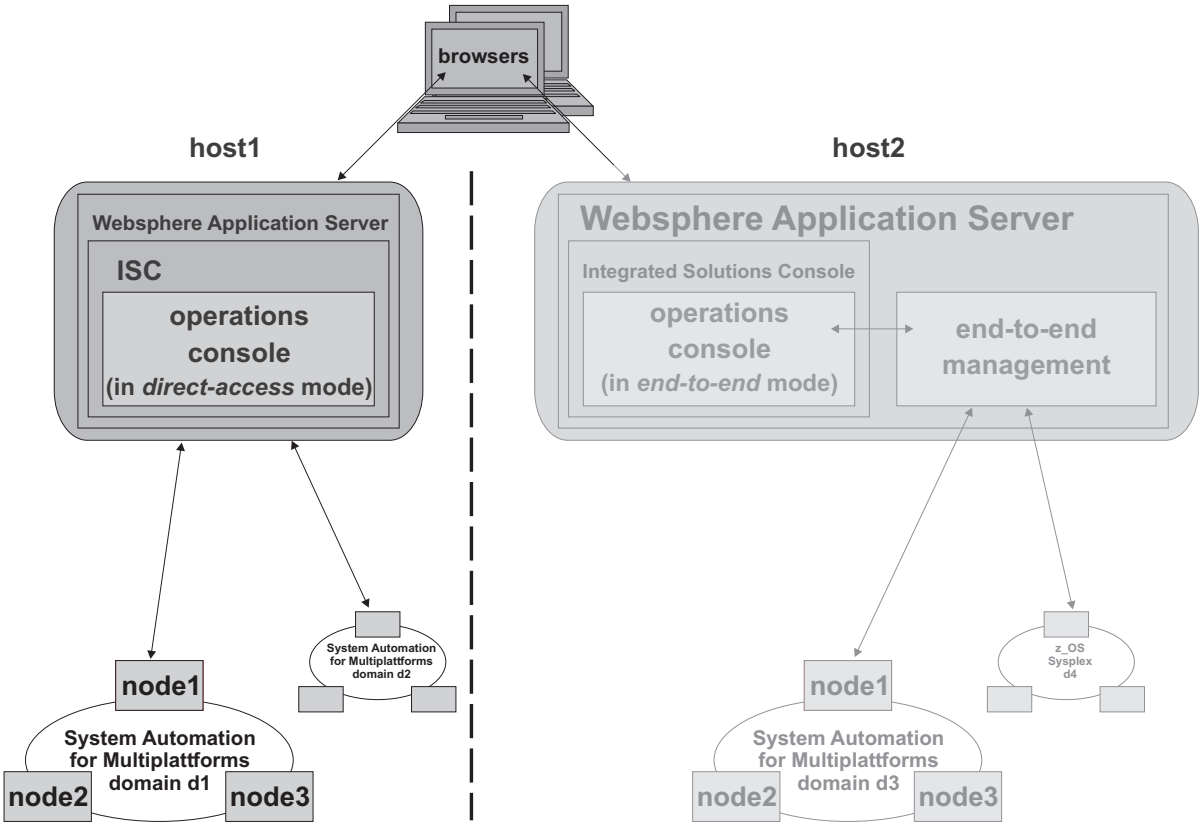



*Figure 5. Overview of the operations console*

The base component of IBM Tivoli System Automation deals with the left part of the figure which shows how the operations console is used in direct access mode without the end-to-end automation part. Direct access mode means that you can only monitor and manage domains controlled by IBM Tivoli System Automation for Multiplatforms. The end-to-end automation management component of IBM Tivoli System Automation for Multiplatforms provides two more modes, the *end-to-end automation mode* and the *first-level automation mode*. Both modes are described in the *IBM Tivoli System Automation for Multiplatforms End-to-End Automation Management Component Administrator's and User's Guide*.
IBM Tivoli System Automation for Multiplatforms domains can be either controlled by the operations console (as shown on the left side of the figure) or by the end-to-end automation management part (as shown on the right side of the figure). Both cannot be done simultaneously.

## Starting and stopping Integrated Solutions Console

To be able to use the operations console and to display the online help for the console, both the Integrated Solutions Console server and the Eclipse Help System server must be started. The following sections describe how you start and stop the servers.

### Starting and stopping the servers on Windows

How you start and stop the Integrated Solutions Console server and the Eclipse Helps System Server on Windows depends on whether or not you are running the servers as Windows services.

## The servers are running as Windows services

If you are running the servers as Windows services, use one of the following approaches:

- You can start and stop the servers from the Windows **Services** panel. These are the relevant entries in the services list:
  - CS01 (ID of the Integrated Solutions Console server)
  - HS01 (ID of the Eclipse Help System server)
- If you want to start or stop the servers from a command prompt when you are running the servers as Windows services, you must start and stop the servers separately. To ensure that the status of the servers is reflected in Windows Services, use the commands described below to start and stop the servers.

  **Note:** Do not use the scripts `StartEclipse.bat` and `StopEclipse.bat` to start or stop the Eclipse Help System server, because then the status of the server will not be reflected in Windows Services.

  **Starting the servers:**

  - To start the Integrated Solutions Console server, use this command:

    ```
    <isc_home>\AppServer\bin\startserver ISC_Portal
    ```

    For example:

    ```
    C:\Program Files\IBM\WebSphere\AppServer\bin\startserver ISC_Portal
    ```

  - To start the Eclipse Help System server, use this command:

    ```
    <isc_home>\PortalServer\ISCEclipse\EclipseServiceStart.bat
    ```

    For example:

    ```
    C:\Program Files\IBM\ISC\PortalServer\bin\EclipseServiceStart.bat
    ```

  **Stopping the servers:**

  - To stop the Integrated Solutions Console server, use this command:

    ```
    <isc_home>\AppServer\bin\stopserver ISC_Portal -username <user_ID> -password <password>
    ```

    where `<user_ID>` and `<password>` are the user credentials of the Integrated Solutions Console administrator.

  - To stop the Eclipse Help System server, use this command:

    ```
    <isc_home>\PortalServer\ISCEclipse\EclipseServiceStop.bat
    ```

    For example:

    ```
    C:\Program Files\IBM\ISC\PortalServer\bin\EclipseServiceStop.bat
    ```

## The servers are not running as Windows services

If you are *not* running the servers as Windows services, use the following commands to start or stop both the Integrated Solutions Console server and the Eclipse Help System server.

To start the servers, use this command:

```
<isc_home>\PortalServer\bin\startISC.bat ISC_Portal
```

For example:

```
C:\Program Files\IBM\ISC\PortalServer\bin\startISC.bat ISC_Portal
```

To stop the servers, use this command:

```
<isc_home>\PortalServer\bin\stopISC.bat ISC_Portal <user_ID> <password>
```

where `<user_ID>` and `<password>` are the user credentials of the Integrated Solutions Console administrator. For example:

```
C:\Program Files\IBM\ISC\PortalServer\bin\stopISC.bat ISC_Portal iscadmin pw4iscadmin
```

## Starting and stopping the operations console on AIX and Linux

To start the Integrated Solutions Console server and the Eclipse Help server, use this command:

```
<isc_home>/PortalServer/bin/startISC.sh ISC_Portal
```

For example:

```
/opt/IBM/ISC/PortalServer/bin/startISC.sh ISC_Portal
```

To stop the Integrated Solutions Console server and the Eclipse Help System server, use this command:

```
<isc_home>/PortalServer/bin/stopISC.sh ISC_Portal <user_ID>  <password>
```

where `<user_ID>` and `<password>` are the user credentials of the Integrated Solutions Console administrator. For example:

```
/opt/IBM/ISC/PortalServer/bin/stopISC.sh ISC_Portal iscadmin pw4iscadmin
```

## Creating and authorizing users and groups

**Note:** Each user who manages first-level automation resources from the operations console must have a user ID on the first-level automation domain. The operations console users can store and change their user credentials for the domains in the credential vault of Integrated Solutions Console (refer to "Managing your user credentials for first-level automation domains" on page 131 for more information).

The installation task created one authorized user (default user ID: iscadmin). To authorize additional users to use the operations console in direct access mode you must perform the following steps:

1. Create the users in Integrated Solutions Console. The tasks you need to perform are described in "Creating users in Integrated Solutions Console."
2. Create a new user group or use an existing user group. The tasks you need to perform are described in "Creating groups in Integrated Solutions Console" on page 112.
3. Assign the new users to the user groups. The tasks you need to perform are described in "Assigning users to groups in Integrated Solutions Console" on page 112.
4. Assign access permissions to the new user group in Integrated Solutions Console. You need to grant the user groups access to the pages of Integrated Solutions Console and to the operations console of IBM Tivoli System Automation. The tasks you need to perform are described in "Assigning access permissions to user groups in Integrated Solutions Console" on page 112.

## Creating users and groups in Integrated Solutions Console

The following sections give an overview of how users and groups are created in Integrated Solutions Console. Additional information is available in the online helps of Integrated Solutions Console. To access the online helps, open the **Help** menu of Integrated Solutions Console and select **Console Basics**. Expand the entry **Settings** page and click **Manage Users and Groups**.

### Creating users in Integrated Solutions Console

To create users, perform the following steps:

1. Log on to Integrated Solutions Console with administrator authority. (Default: ISC user: iscadmin, ISC group: iscadmins). Use the user ID that was created during the installation of the operations console.

   ---

2. Click the **Console Settings** tab to open the Settings page.

   ---

3. Select the menu entry **User and Group Management** to display the Manage Users and Groups page.

   ---

4. In the table on the page, click **all authenticated portal users**.

   ---

5. Click **New user**.

   ---

6. Enter the user ID and password, and the user's first name, last name, and e-mail address, and click **OK**.

Repeat these steps starting from step 4 on page 111 for each user you want to authorize.

### Creating groups in Integrated Solutions Console

To create groups, perform the following steps:

1. Log on to Integrated Solutions Console with administrator authority. (Default: ISC user: iscadmin, ISC group: iscadmins)

   Use the user ID that was created during the installation of the operations console.

2. Click the **Console Settings** tab to open the Settings page.

3. Select the menu entry **User and Group Management**.

4. Click **New group**.

5. Type the name of the user group you want to create and click **OK**.

### Assigning users to groups in Integrated Solutions Console

To assign the users to the groups, perform the following steps:

1. Log on to Integrated Solutions Console with administrator authority. (Default: ISC user: iscadmin, ISC group: iscadmins). Use the user ID that was created during the installation of the operations console.

2. Click the **Settings** tab to open the **Settings** page.

3. Select the menu entry **User and Group Management**.

4. Select the table entry **all portal user groups**.

5. Select the group to which you want to assign users and click **Add member**.

6. Select the users you want to add to the group and click **OK**.

Repeat these steps until you have assigned users to all of the groups.

## Assigning access permissions to user groups in Integrated Solutions Console

After creating the user groups in Integrated Solutions Console, you must perform the following tasks:

- "Granting user groups access to the pages of Integrated Solutions Console" on page 113.

- "Granting user groups access to the operations console of IBM Tivoli System Automation for Multiplatforms" on page 114.

## Granting user groups access to the pages of Integrated Solutions Console

Perform the following steps:

1. Log in to Integrated Solutions Console as administrator (default: user ID iscadmin, group iscadmins)

   _____

2. In the navigation tree of Integrated Solutions Console, expand **Console Settings**.

   _____

3. Click **Resource Permissions** to display the **Resource Types** list.

   _____

4. In the **Resource Types** list, click **Pages** to display the list of resources.

   _____

5. In the **Resources** list, click  for **Content Root**.

   _____

6. In the **Roles** list, click  for **User**.

   _____

7. Click **Add**. On the page that appears, only the groups are listed for which access has already been granted. If you perform this task for the first time, the page may be empty.

   _____

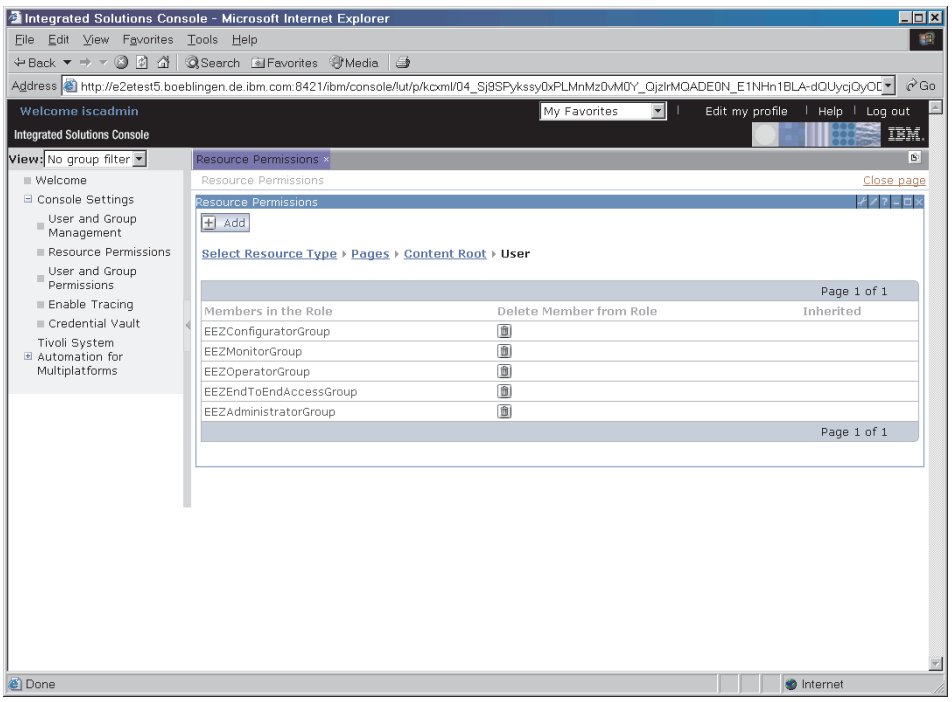
8. Click **Search**. (In the field **Search for Users or User Groups**, the entry **User Groups** must be selected.)

   _____

9. In the **Users and User Groups** list, select the check boxes for the automation-specific groups:

   _____

10. Click **OK**. The following panel appears:




11. Restart Integrated Solutions Console.

## Granting user groups access to the operations console of IBM Tivoli System Automation for Multiplatforms

Perform the following steps:

1. Log in to Integrated Solutions Console as administrator (default: user ID iscadmin, group iscadmins)
2. In the navigation tree of Integrated Solutions Console, expand **Console Settings**.
3. Click **Resource Permissions** to display the **Resource Types** list.
4. In the **Resource Types** list, click **Portlet Applications** to display the list of resources.
5. In the **Resources** list, click  for **Tivoli System Automation for Multiplatforms Operations Console**.
6. In the **Roles** list, click  for **User**.
7. Click **Add**.
8. Click **Search**. (In the field **Search for Users or User Groups**, the entry **User Groups** must be selected.)

9. In the **Users and User Groups** list, select the check boxes for the automation-specific groups:

10. Click **OK**.

11. Restart Integrated Solutions Console.

## Modifying and deleting users and groups

The following sections describe how to modify users and groups.

### Changing passwords for users on Integrated Solutions Console

Perform the following steps to change user passwords:

1. Log on to Integrated Solutions Console as administrator (default: user ID iscadmin, group iscadmins). Use the user ID that was created during the installation of the operations console.

2. Open the Settings page.

3. Select the menu entry **User and Group Management**.

4. Select the menu entry **all authenticated portal users**.

5. Click the **Edit** button for the user ID you want to modify.

6. Type the new password in the entry fields.

7. Click OK.

### Deleting user IDs on Integrated Solutions Console

Perform the following steps to delete users:

1. Log on to Integrated Solutions Console as administrator (default: user ID iscadmin, group iscadmins). Use the user ID that was created during the installation of the operations console.

2. Open the Settings page.

3. Select the menu entry **User and Group Management**.

4. Select the menu entry **all authenticated portal users**.

5. Click the **Delete** button for the user ID you want to delete.

6. Click **OK**.

## Deleting groups on Integrated Solutions Console

Perform the following steps to delete a group:

1. Log on to Integrated Solutions Console as administrator (default: user ID iscadmin, group iscadmins).Use the user ID that was created during the installation of the operations console.
2. Open the Settings page.
3. Select the menu entry **User and Group Management**.
4. Select the menu entry **all portal user groups**.
5. Click the **Delete** button for the group you want to delete.
6. Click **OK**.

## Configuring your Web browser

To be able to display the operations console in you Web browser, the following settings are required:

- JavaScript must be enabled in all Web browsers.
- For Microsoft Internet Explorer, the following settings are required:
  - Set the security level to medium.

    Do not set the security level to high. If high security is required, ensure that the entry **ActiveX controls and plugins - Initialize and Script ActiveX controls not marked as safe** on the Security settings page is set to **Enable**. Otherwise, the information displayed in the operations console is not updated automatically.
  - Set **Scripting - Active Scripting** to **Enable** on the Security settings page. Otherwise, navigating the operations console is not possible.

## Logging on

To access the operations console, you have to perform the following steps:

1. Open Integrated Solutions Console in a Web browser window.
2. Log on to Integrated Solutions Console using your user ID and password.
3. Connect to the operations console.

The following section describes how to log on to Integrated Solutions Console and connect to the operations console.

**Note:** It is recommended that you do not use multiple browser windows on the same client system simultaneously to connect to the same Integrated Solutions Console, because browser types other than Microsoft Internet Explorer will share a single HTTP session between multiple browser instances if these instances are running on the same system and connect to the same Integrated Solutions Console.

Working with multiple browser instances using the same HTTP session will cause unexpected results. The same situation occurs if you open multiple Microsoft Internet Explorer browser windows using **File —> New Window** (or Ctrl+N) from an existing Integrated Solutions Console session, because in this case the new browser window and the one from which it was opened will also share the same session.

### Steps for accessing the operations console

To access the operations console, perform the following steps:

1. Open a Web browser window and type the address of Integrated Solutions Console in the **Address** field. The entry must have the following form:

   `http://<hostname>:<port>/ibm/console`

   In this entry, `<hostname>` is the name of the host and `<port>` is the port Integrated Solutions Console runs on. The default port is 8421.

   The log in panel of Integrated Solutions Console is displayed in the browser window:

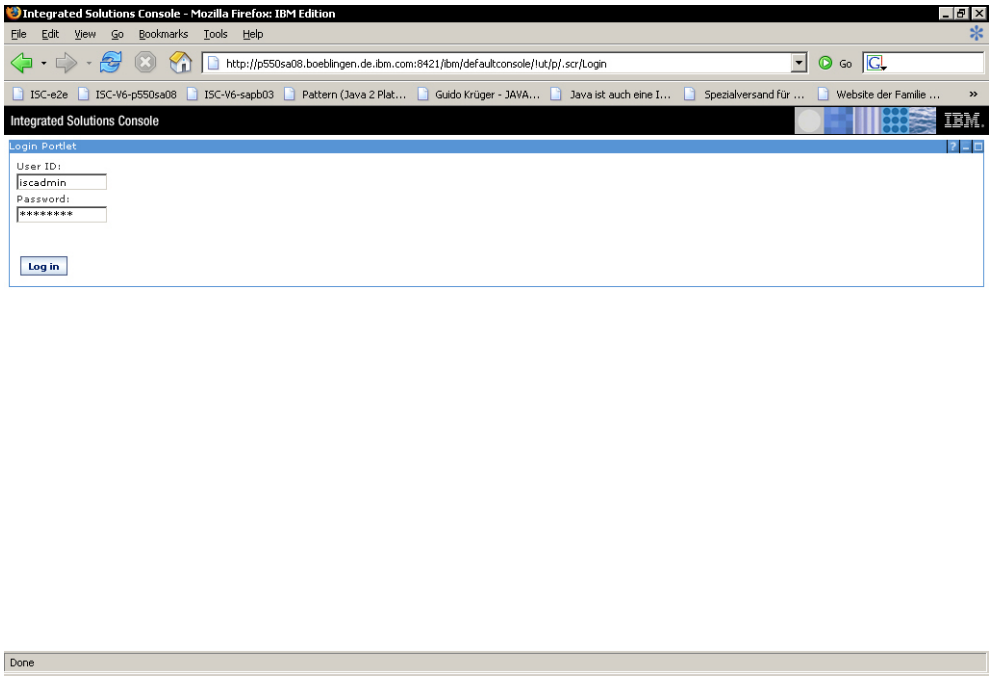



*Figure 6. Log in panel of Integrated Solutions Console*

2. Specify your user ID and password and click **Log in**. The Welcome page of Integrated Solutions Console comes up. On the Work Items page, the suites that are installed are listed. For each suite, the suite name and version number are listed:
3. In the navigation tree on the left, expand the folder **Tivoli System Automation for Multiplatforms**.
4. Click **SA operations console**.
5. The main panel of the operations console is displayed.

# Understanding the layout of the operations console

The main panel of the operations console of IBM Tivoli System Automation is divided into several areas:

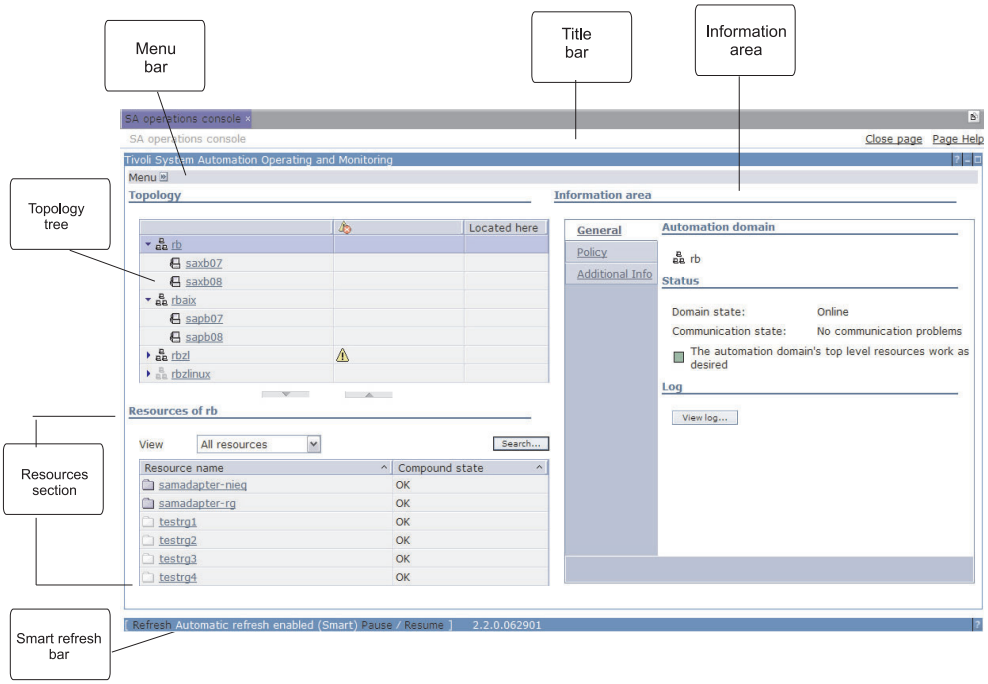



*Figure 7. Main panel of the operations console*

**Title bar**

Use the controls on the title bar to display the online help page for the panel you are displaying and for minimizing and maximizing the panel.

**Menu bar**

Use the entries in the Menu which is available on the menu bar to update the information displayed in the topology tree and the resource table, to change your user preferences, and to display information about the version of the operations console you are using.

**Information bar**

The information bar is not shown in the figure above. It is displayed below the menu bar when you have performed an action on an element in the operations console. It displays a message confirming that the request or command has been submitted for processing. The message on the information bar only confirms the initial action, it is not updated while the command or request is being processed. The results of the system actions that are performed due to the request or command are reflected in the operations console itself. There you can see, for example, that the status of a resource has changed.

The confirmation message is replaced with a new message whenever you perform an action against an element in the operations console. Clicking **Clear** on the information bar hides the information bar from view. It reappears with a new confirmation message when you perform an action on an element.

**Topology tree**

The topology tree shows the automation domains and the nodes that

belong to the domains. The topology tree displays state-related information, allows you to select and work with domains and nodes, and is used to control what is displayed in the resource table. For more information, refer to "What you must know about the topology tree."

**Resources section**
Use the areas of the resources section to work with resources:

**View and Search**
The View and Search functions allow you to limit the scope of the resource table. **View** allows you to display only resources that are in an error or warning state. **Search** allows you to display only resources that meet specific search criteria.

**Resource table**
Displays a list of resources and their states. You use it to select and work with resources. The resource table has two views:

**Search results view**
When you use **Search** to see only a specific set of resources in the resource table, the search results are displayed in the search results view. For more information, refer to "Search results view" on page 126.

**Group hierarchy view**
The group hierarchy view is displayed when you are not displaying the results of a search. For more information, refer to "Group hierarchy view" on page 124.

For more information about the resources section, refer to "What you must know about the resources section" on page 123.

**Information area**
Use the pages in the information area to obtain information about the element you have selected in the topology tree or resource table, and to perform actions against the element. For more information about the information area, refer to "What you must know about the information area" on page 129.

**Smart refresh bar**
On the Smart refresh bar, you can invoke an immediate smart refresh of the operations console, suspend and reactivate the smart refresh function, and you can see whether smart refresh is enabled or suspended.

## What you must know about the topology tree

The **Topology** tree has three columns:

- The Topology column shows the automation domains  and the nodes  that belong to a domain in a hierarchical view.
- The **Status** column shows the health status of the domain.
- The **Located here** column is used to identify by which domain a resource is hosted and on which node or nodes it is located.

### Navigating the topology tree

You click the twistie in front of a domain icon to expand or collapse the nodes belonging to the domain.

## Selecting an element in the topology tree

To select an element in the tree, click its name. When you select a domain or node, you influence what is displayed in the resource table and in the information area:

- The resource table shows the resources that are hosted by the selected domain or that are located on the selected node.
- The pages in the information area show information about the element that is selected in the topology tree. Depending on which type of element you have selected, buttons are enabled on the pages that let you perform actions against the element.

## Limiting the scope of the topology tree

By default, all automation domains are displayed in the topology tree. When you are not interested in seeing all automation domains, you can hide domains from view. To limit the scope of the topology tree, you use the Visible domains page of the Preferences panel. To open the Preferences panel, click **Menu** in the menu bar and select **Preferences**.

## What is displayed in the topology column

In the topology column you see the automation domains and the nodes that are managed by each automation domain.

The following icons are used to identify the elements of the topology tree:

*Table 15. Icons used for the elements of the topology tree*

| Icon | Description |
|---|---|
|  | An automation domain. When the domain is not online or its state is unknown, the icon is grayed-out. |
|  | A node that belongs to an automation domain. When a node is not online, the icon is grayed-out. |

## What you can see in the Status column

The Status column is used to inform you of the health status of a domain. When the domain is healthy, the column is empty.

A domain is considered healthy if none of the resources that you are using as domain health indicators has encountered a problem that may require your attention. By default, the top-level resources of a domain are used as domain health indicators. However, on the Visible domains page on Preferences panel (**Menu –> Preferences**) you can define that a different set of resources is to be used for the purpose.

When a resource that is used as domain health indicator has a problem, one of the following icons appears in the Status column:

*Table 16. Icons in the Status column of the topology tree*

| Icon | The icon indicates ... |
|---|---|
|  | A warning has been issued. The problem may still be solved automatically, but the element should be monitored carefully. |
|  | The red error icon indicates that an error has occurred. To resolve the error, operator intervention is required. |

*Table 16. Icons in the Status column of the topology tree (continued)*

| Icon | The icon indicates ... |
|---|---|
|  | The black error icon indicates that an unrecoverable error has occurred. To resolve the problem, urgent operator intervention is required. |

**Note:** If you see a warning icon right beside the domain name, this may indicate that the user ID and password for this IBM Tivoli System Automation domain has not been entered. Double click on the domain name and you are prompted for entering this user ID and password. Your system administrator may help you in getting the user ID and password.

## What you can see in the Located here column

You use the **Located here** column to find out to which domain or domains a resource or group of resources belongs and on which nodes they are located. To determine the location of a resource or resource group, you select the resource or resource group in the resource table. When you have made your selection, a check mark appears in the **Located here** column for the domain that hosts the resource. If you are displaying the node hierarchy, a check mark will identify the node or nodes on which the resource is located.

## What you must know about the resources section

The following figure shows the layout of the resources section.

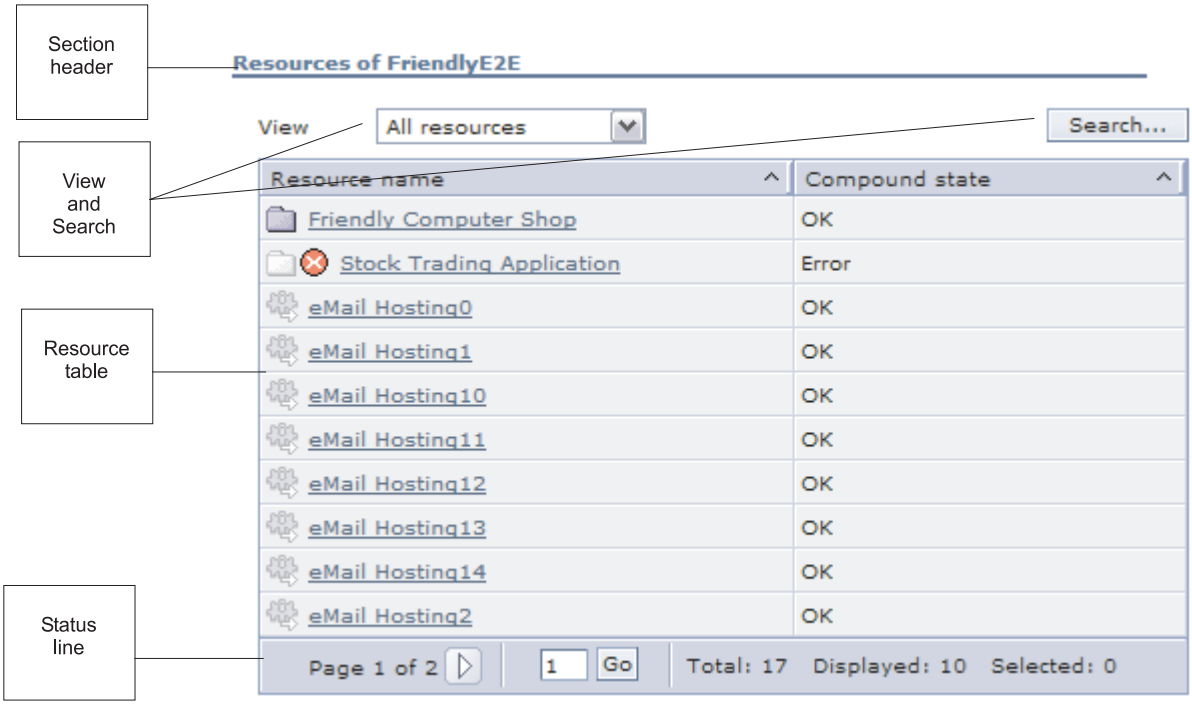



*Figure 8. Layout of the resources section*

The resources section has the following areas:

### Section header

The section header displays the name of the domain or node that is currently selected in the topology tree.

### View and Search

The View and Search functions allow you to limit the scope of the resource table:

**View** Select the **Errors and warnings** item from the View drop-down list to display only resources that are in an error or warning state. The view is always applied to the list of resources which is currently displayed in the resource table.

**Search**
Allows you to display only resources that meet specific search criteria.

### Resource table views

The resources which you are monitoring and managing from the operations console are displayed in the resource table. The resource table has two views, which are described in the sections below. In both views, you can perform the following basic actions:

**Select a resource**
To select a resource, you click its name in the resource column.

**Control the sort order of the resource table**

You can sort the resource table on any column by clicking the sort arrow in the column header.

A solid sort arrow in a column header indicates that the table is currently sorted on the column. The direction in which the solid sort arrow is pointing indicates the current sort order (ascending or descending). By clicking on the solid sort arrow, you can toggle between ascending and descending sort order.

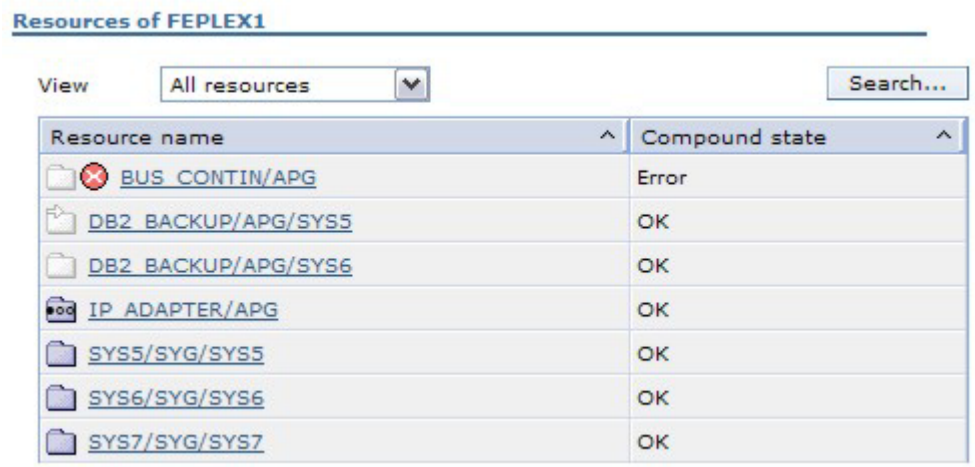
When you position the cursor over a sort arrow, a hover help text appears showing the current sort status of the column and the sort order that will result when the sort arrow is clicked.

**Page through the resource table**

The resource table may extend over multiple pages. To page through the table or to navigate to a specific page you use the controls that are available in the status line below the table.

## Group hierarchy view

The group hierarchy view is displayed when you are not displaying the results of a search. In the following figure, the top-level resources of the automation domain "FEPLEX1", which is selected in the topology tree, are displayed in the resource table.

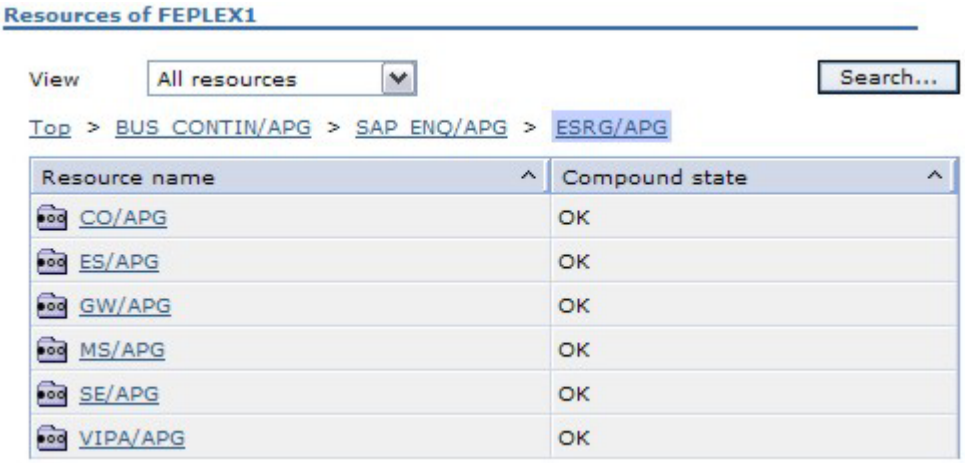



**Resource column:**
In this column, you see a list of resources and their states. Which resources are listed depends on which selection you have made:

- A domain or node is selected in the topology tree: The top-level resources of the domain or node are displayed.
- A group is selected in the resource table: The members of the group are displayed.

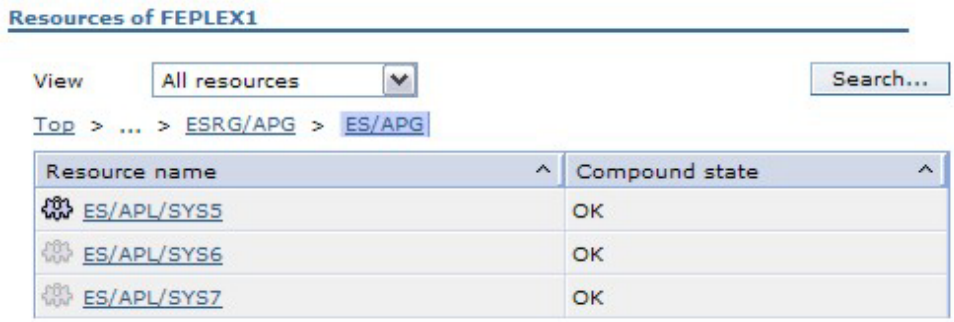
To sort the resources alphabetically by name, click the sort arrow in the column header.

When you select a group in the resource table, the members of the group are displayed in the resource table. In the area above the table, a bread crumb trail appears. On the trail, the name of the group whose members are listed in the resource table is highlighted, indicating that the group is selected.




The bread crumb trail is useful for both navigation and orientation:

- When you drill down into the group hierarchy, an entry is added to the trail for each group you select.
- The last entry on the trail identifies the group whose members are currently displayed in the resource table. When the group name is highlighted, the group is selected and the group details are displayed in the information area.
- When you click **Top** on the bread crumb trail, the top-level resources of the automation domain or node that is selected in the topology tree are again displayed in the resource table and the bread crumb trail disappears.
- When the bread crumb trail starts to get deeper than three levels, an ellipsis symbol (...) replaces all but the last two entries on the trail.




The ellipsis symbol cannot be clicked. To navigate upward through the group hierarchy, click an available group name on the trail until the group you want to view appears again.

The resource icon to the left of the resource name in the resource column indicates both the resource type and its online status:

- When the resource is online, its icon is active, when the resource is offline, the icon is grayed out.
- When a resource is in a warning or error state, the resource icon is highlighted with a warning or error icon.
- An operator icon indicates that an operator request was submitted against the resource. The color of the operator icon changes while the request is being

processed, yellow indicates that the request has been submitted, green indicates that the request was completed successfully.

The following types of resources are displayed in the resource column:

*Table 17. Resource icons in the resource column*

| Icon | Description |
|---|---|
|  | A resource |
|  | A resource group |
|  | A move group, also called a floating resource. |

The following table lists the warning and error icons that appear in the resource column when a resource is in an error or warning state.

| Icon | Description |
|---|---|
|  | The yellow warning icon indicates that the resource is in warning state. The problem may be resolved automatically but the resource should be monitored carefully. |
|  | The red error icon indicates that the resource is in an error state. Operator intervention is required. |
|  | The black error icon indicates that the resource has encountered an unrecoverable error. Immediate operator intervention is required. |

**Compound state column:**
The column shows the compound state of the resource. By sorting on this column, you can group the resources by state.

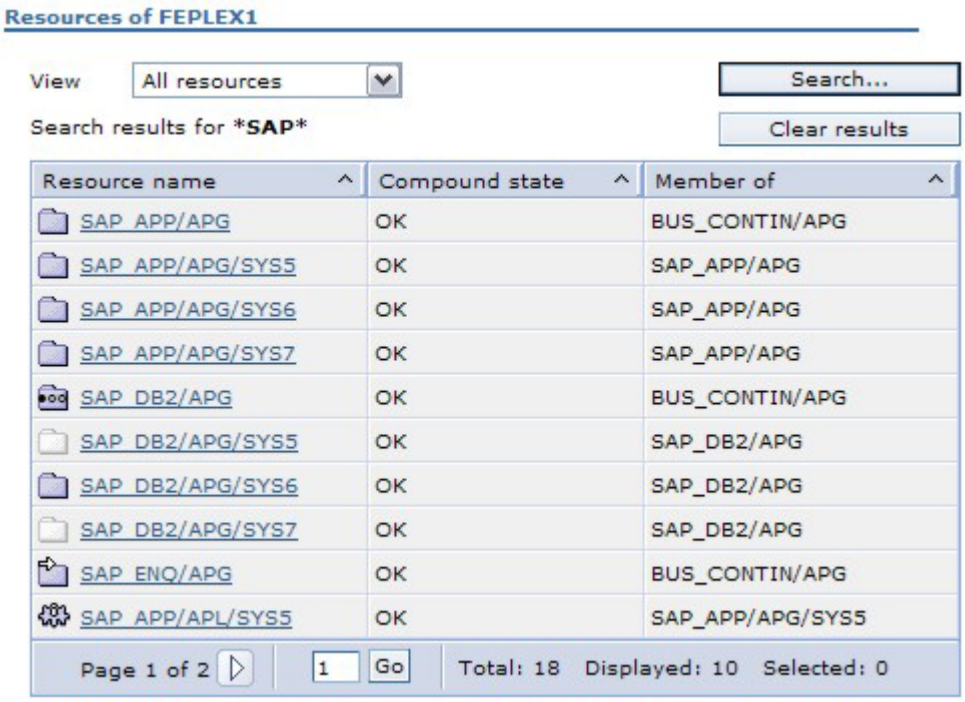
The compound state can have one of the following values:

| State | Description |
|---|---|
| OK | The resource is working as desired. Select the resource to display more information. |
| Warning | The resource is in warning state. Select the resource to display more information. |
| Error | The resource is in an error state. Select the resource to display more information. |
| Fatal | The resource has encountered an unrecoverable error. Select the resource to display more information. |

## Search results view

When you use **Search** to see only a specific set of resources in the resource table, the search results are displayed in the search results view. In the area above the

resource table, the search criteria that were used for the search are displayed. In this view, the resource table has the following layout:




To limit the scope of resources that are currently displayed in the resource table to those that are in an error or warning state, you can additionally apply the **Errors and warnings** view that is provided in the **View** field.

**Resource table columns:**
In the search results view, the resource table has three columns:

**Resource column**

In the column, the resources that match the search criteria are listed.

- To sort the resources alphabetically by name, click the sort arrow in the column header.
- If a resource is in a warning or error state, the resource icon is highlighted with a warning or error icon.
- If an operator request was submitted against the resource, an operator icon is displayed.
- Clicking a resource selects the resource and its details are displayed in the information area.

  **Note:** When you select a group in the search results view, the group details will be displayed in the information area, but the resource table will not switch to the group hierarchy view to display the group members.

  To display the group's members in the group hierarchy view, you must select the group and click **Clear results** (see “Clearing the search results” on page 128).

**Compound state column**

The column shows the compound state of the resource. By sorting on this column, you can group the resources by state.

**Member of column**

If a resource is a member of a group, the name of the group is displayed in this column. When you sort the resource table on this column, the resources that are members of the same group are listed next to each other.

**Clearing the search results:**

When you click **Clear results**, the resource table switches back to the group hierarchy view. Which resources are then displayed in the group hierarchy view, depends on your selection in the search results view:

- No resource was selected: The top-level resources of the automation domain or node that is selected in the topology tree are displayed.
- A resource group was selected: The group members are displayed. On the bread crumb trail, the name of the group is highlighted, the group details are displayed in the information area.
- A resource that is a member of a group was selected: The group members are displayed, the group name is displayed on the bread crumb trail but is not highlighted, the name of the selected resource is highlighted in the resource list.

## What you must know about the information area

This section gives a short overview about the pages in the information area.

In the information area you find detailed information about the element that is currently selected in the topology tree or in the resource table. On the pages in the information area, controls are available that let you perform actions on the selected resource, group, node, or automation domain. Which pages are displayed and what they contain depends on the type of element that is currently selected in the topology tree or resource table:

| When you select ... | ...these pages are available |
|---|---|
| an automation domain in the topology tree | • General<br>• Policy<br>• Additional Info |
| a node in the topology tree | • General<br>• Additional Info (available only if additional information exists) |
| a resource or a resource group in the resource table | • General<br>• Relationships (available only if the resource has relationships)<br>• Additional Info (available only if additional information exists) |

## General page

The General page is always available when an element is selected in the topology tree or the resource table. Which information is provided on the General page and which actions are possible depends on the selection you have made in the topology tree or the resource table.

**General page for a domain**
Use the General page for a domain to get information about the state of the domain, its communication state, and the state of the resources it hosts and to display the log file for the domain.

**General page for a node**
Use the General page for a node to get detailed information about the node, for example, its name, class, and possibly, a description of the node. In addition, the page contains information about the observed state of the node. A button is provided that lets you exclude the node from automation or include it in automation.

**General page for a resource group**
Use the General page for a resource group:

- to get more detailed information about the resource group itself, for example, its class.
- to display detailed information about the different states of the group.
- to submit a start or stop request.
- to view the request stack of the resource.
- to find out if the resource group is a member of any other resource group. If this is the case, a link is provided that lets you jump to that resource group.

**General page for a resource**

Use the General page:

- to display detailed information about the resource, for example, its class and owner.
- to obtain detailed information about the different states of the resource.
- to view the operator requests if there are any.
- to submit a start or stop request.
- to find out whether the resource is a member of a group.

## Policy page

The Policy page is available when a domain is selected in the topology tree. It shows the policy name and its activation time and date.

## Additional Info page

An Additional Info page can be displayed for automation domains, nodes, resources, and resource groups when additional information is available.

## Relationships page

A Relationships page is available when you have selected a resource or group in the resource table and backward or forward relationships have been defined for the resource.

## Managing your user credentials for first-level automation domains

You can store the user IDs and passwords you need for logging on to a first-level automation domain in the Integrated Solutions Console credential vault. From there, the information is retrieved automatically when needed. This saves you from having to enter your user ID and password in these cases.

Perform the following steps to manage your user credentials for first-level automation domains:

1. Open the Preferences panel (**Menu —>Preferences**).

2. Open the User IDs page.

3. Select a domain from the list of domains.

4. You have the following options:
   - To add or change your credentials, click **Edit**. The Credentials panel is displayed.

     On the Credentials panel, create or change your user credentials. To store the credentials in the credential vault, ensure that the check box on the panel is selected.
   - To delete a specific set or all of your credentials, click **Delete** or **Delete all**.

# Managing resources using the operations console

In direct access mode, managing resources means:

- Starting or stopping a resource or a group of resources.
- Excluding a node from automation and including it again.

## Working with requests

You start and stop resources by changing their desired state. You can achieve this by submitting start or stop requests that bring a resource online or offline. The desired state of a resource will be changed when your request wins. The resource will only be started or stopped after all relationships have been fulfilled. See the description in "Using requests to start and stop resource groups and resources" on page 170.

For submitting requests, the following rules apply:

- Online requests can only be submitted against resources in desired state Offline.
- Offline requests can only be submitted against resources in desired state Online.
- Requests cannot be submitted if the current desired state of the resource results from an operator requests. In this case, the operator requests must be canceled.

**Note:** Due to the internal design of the base component, requests against a resource group are not propagated to members that are floating resources.

### Submitting start requests

Perform the following steps to submit a start request:

1. In the resource table, select the resource you want to start.

2. On the **General** page, click **Request Online**.

   The Request Online panel is displayed.

3. Use the entry field on the Request Online panel to provide a short description of why you want to change the automation goal of the resource to Online.

4. Click **Submit** to submit the request.

**Results:**

- A confirmation message is displayed on the information bar, indicating that the request has been submitted for processing.
- After the next refresh, the icon of the resource is changed to indicate that a request has been issued against the resource.
- The request is processed. Processing of the request is complete when the resource has been started. The operator icon turns green.

### Submitting stop requests

Perform the following steps to submit a stop request:

1. In the resource table, select the resource you want to stop.

2. On the General page, click **Request Offline**. The button is only enabled if the resource's desired state is Online and no other operator requests against the resource exist.

   The Request Offline panel will be displayed.

3. Use the entry field to provide a short description of why you want to change the automation goal of the resource to Offline.

4. Click **Submit** to submit the request.

**Results:**

- A confirmation message is displayed on the information bar, indicating that the request has been submitted for processing.
- After the next refresh, the icon of the resource is changed to indicate that a request has been issued against the resource.
- The request is processed. Processing of the request is complete when the resource has been stopped. The operator icon turns green.

## Displaying information about an operator request

When an operator has submitted a start or stop request against a resource, an operator request icon appears on the General page for the resource. The icon indicates whether the request has been completed successfully or not:

*Table 18. Operator request icons in the information area*

| Operator request icon | Description |
|---|---|
|  (yellow) | A stop request has been submitted. The yellow operator icon indicates that the observed state of the resource is not Offline yet. |
|  (yellow) | A start request has been submitted. The yellow operator icon indicates that the observed state of the resource is not Online yet. |
|  (green) | The green operator icon indicates that the stop request has been completed successfully. The observed state of the resource is Offline. |
|  (green) | The green operator icon indicates that the start request has been completed successfully. The observed state of the resource is Online. |

This is how you can display more information about the request:

- Move the mouse over the operator request icon to display the user ID of the operator who submitted the request.
- Click the operator request icon to bring up the request details panel.

## Displaying request lists

All requests and votes that have been submitted against a resource are added to the resource's request list. You can display the list to find out which requests have been issued and which of the requests wins. The list is sorted by priority with the winning request listed at the top.

The list contains information about each request or vote, for example:

- its source (for example, the name of the operator who submitted the request)
- its priority
- the creation date and time

From the Request list panel, you can display detailed information about each of the requests or votes, including the comments that were added by operators when they submitted the request. Note that, due to the internal design of the base component, requests against a resource group are not propagated to members which are floating resources.

**Steps for viewing a request list and the request details:**
Perform the following steps:

1. In the resource table, select the resource whose request list or request details you want to view.

---

2. On the General page, click **View requests**.

   The Request list is displayed. The list is sorted by priority. The first entry is the winning request.

---

3. To display the details for a request, select the resource in the list and click **More info**.

   The Request details panel is displayed.

---

## Canceling requests

You can cancel operator requests that have been submitted against resources. Votes and requests generated by automation managers cannot be canceled.

This is what happens when you cancel a request:

- When you cancel a request that did not win, you prevent it from being completed at a later time.
- When you cancel the request that is responsible for the current desired state of the resource, you change the desired state of the resource to the opposite if there are no other requests or votes in the request list that will win when the canceled request is removed.
- When you cancel a request, votes that were generated against other resources because of StartAfter or StopAfter relationships are canceled as well.

**Steps for canceling requests:**
Perform the following steps to cancel a request:

1. Select the resource in the resource table.

---

2. On the **General** page, click **Cancel request**.

   The button is enabled only if there is an operator request in the request list of the resource.

   The text to the left of the **Cancel request** button describes the resource's probable desired state after the request has been canceled. The probable desired state is calculated as follows:

- If there are other requests or votes in the request list, the winning request determines the expected desired state.
- If there are no other request or votes in the list, the desired state that is defined in the policy becomes the automation goal.
- If there are other requests or votes in the request list, the one with the highest priority will win.

The desired state that is actually set after the cancelation can differ from the expected state, for example, when a new request or vote is generated at the same time or immediately after you canceled the request.

---

# Chapter 9. Protecting your resources – quorum support

This chapter describes how IBM Tivoli System Automation protects your resources by using configuration and operational quorum.

## Overview

A cluster (also known as peer domain) may split into two or more subclusters in case no more communication is possible between the elements in the cluster. Since each subcluster is not aware of one another, it may occur that IBM Tivoli System Automation starts a new instance of an application that is already running in one of the other subclusters. If the application requires access to a shared disk, for example to perform failure recovery, data corruption may occur due to simultaneous access to the disk.

Such resources are characterized as critical. A critical resource is a resource that may not be running on more than one node at any point in time. If such a resource is active on two or more separated nodes, then data integrity of the cluster is endangered. In order to protect such critical resources, IBM Tivoli System Automation ensures that only one of the subclusters survives while the others are dissolved. Thus IBM Tivoli System Automation prevents data corruption that is caused by system failures or network partitions.

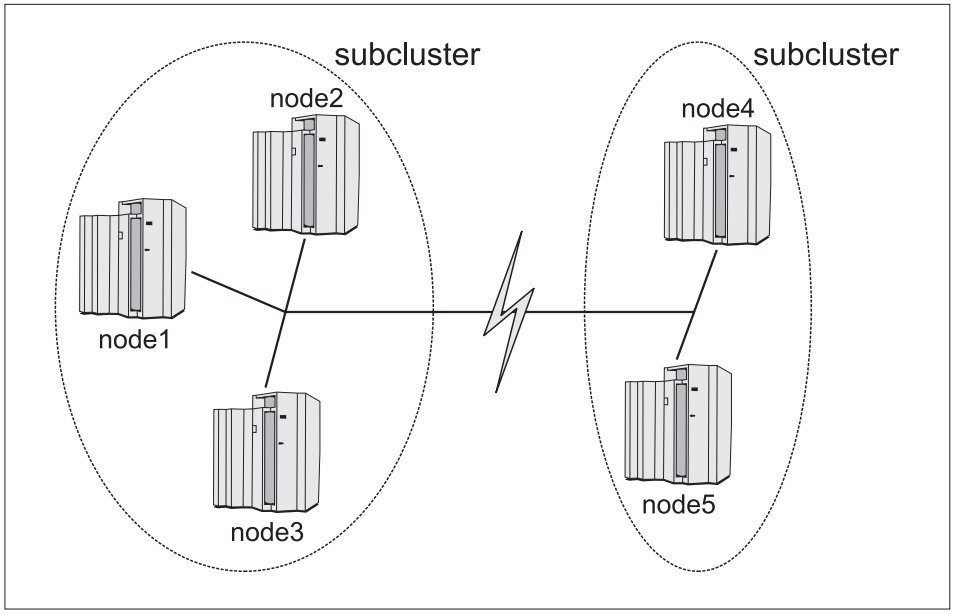



*Figure 9. Quorum – majority of nodes*

If a cluster falls apart into two or more subclusters, the configuration resource manager (ConfigRM) determines which of the subclusters has the majority of nodes. The majority is given when the subcluster has more than half of all defined nodes in the cluster. The subcluster with the majority of nodes will have an operational quorum. It will survive and become the active cluster, while the other subcluster(s) will be dissolved.

The protection of critical resources is achieved by

- Configuration quorum
- Operational quorum

## Configuration quorum

Configuration Quorum determines when configuration changes in the cluster will be accepted. The integrity of the cluster definition is ensured by following the majority rule. Operations affecting the configuration of the cluster are only allowed when n/2+1 nodes are active, where n is the number of nodes defined in the cluster. However, for some operations, the majority rule can be overridden or different configuration quorum rules apply:

- You can remove nodes using the **rmrpnode** command if exactly half of the nodes is online and if the configuration can be successfully removed from at least one of the offline nodes. You can also use the -f option of this command to override the majority rule.
- The quorum rule for the **startrpdomain** command is n/2, but you can override it with the all nodes (-A) option or the local node (-L) option.

**Note:** In a tie situation you can start and stop resource groups using the command **chrg –o** *online/offline group_name*.
For more details refer to the *IBM Reliable Scalable Cluster Technology for Linux, Administration Guide*, SA22-7892, and *IBM Reliable Scalable Cluster Technology for Linux, Technical Reference*, SA22-7893.

## Operational quorum

The operational quorum is used to decide whether resources can be safely activated without creating conflicts with other resources. Operational quorum is determined based on the number of online nodes and a tie breaker to resolve certain tie situations. A subcluster has an operational quorum if it has more than half of the nodes active.

If an operational quorum exists, IBM Tivoli System Automation can manipulate resources or resource groups and bring them online. If no quorum exists, IBM Tivoli System Automation cannot take any action on a resource.

If critical resources are active on a subcluster that has lost quorum, the ConfigRM uses the ″CritRsrcProtMethod″ attribute of each node in the subcluster to determine on which way the system should be terminated. The protection methods are based on immediate system shutdown by means of kernel panic simulation. There are 6 protection methods:

| Meaning | Value |
|---|---|
| Hard reset and reboot operating system (default). | 1 |
| Halt operating system. | 2 |
| Hard reset and reboot operating system with Sync. | 3 |
| Halt with Sync. | 4 |
| No protection. System continues operating. | 5 |
| Exit and restart RSCT subsystems. | 6 |

A protection method with sync sends file system buffers to disc before the operating system is halted or reset. Thus the probability of data loss or data

inconsistency on the file system is reduced. Note that protection methods with sync might also be an unsafe solution in certain situations. There might be a chance that the data is already being accessed during file system flush from another application constituent that has just started in the subcluster which gained operational quorum. It must be examined if this is likely to occur in a given system and application environment.

Independent of the protection method being used, it is highly recommended to use a journaling file system. This can prevent the corruption of the file system itself and greatly enhances file system recovery after a system reset.

The protection methods on the nodes may be different. However, the normal case is that the same protection method is set for each node in the entire cluster.

In the case of a tie in which the cluster has been partitioned into subclusters with an equal number of nodes, the configuration resource manager uses a tie breaker to determine which subcluster has an operational quorum. A subcluster will have an operational quorum if it has the tie breaker reserved.

The following tie breaker types exist:

1. Operator - this tie breaker asks the operator or administrator for a decision. Until the administrator explicitly breaks the tie, neither subcluster will have an operational quorum. The operational quorum state is set to ″PendingQuorum″ and stays in this state until either the network is repaired, failing nodes are repaired and brought online, or the operator determines which subcluster is the winning one and which is the losing one. This is done by invoking the ″ResolveOpQuorumTie″ action on a node of each active subcluster.
2. Fail - this actually is a pseudo tie breaker which means that it does not resolve the tie. Neither subcluster will have operational quorum.
3. SCSI - this tie breaker is specific for Linux on System i, Linux on System p, and Linux on System x. It assumes that an SCSI disk is shared by all nodes of the cluster. Tie breaker reservation is done by the SCSI reserve or persistent reserve command.
4. ECKD - this tie breaker is specific for Linux on System z. It assumes that an ECKD disk is shared by all nodes of the cluster. Exclusive access to the disk is done by means of the ECKD reserve command.
5. DISK - this tie breaker type is specific for AIX. This tie breaker type enables you to specify an SCSI or SCSI-like physical disk using an AIX device name, and assumes that the SCSI disk is shared by one or more nodes of the cluster. Tie breaker reservation is done by the SCSI reserve or persistent reserve command. If creating a tie breaker of this type, you need to set the DeviceInfo persistent resource attribute to identify the physical disk. Only SCSI and SCSI-like physical disks are supported. Physical disks attached via Fiber Channel, iSCSI and Serial Storage Architecture Connections are suitable.
6. EXEC - this tie breaker is a generic tie breaker implementation which calls a custom executable for tie breaker operations. The network tie breaker (samtb_net) shipped with this product is implemented as an EXEC tie breaker executable. See “Network tie breaker” on page 150 for details how to configure the EXEC tie breaker to use samtb_net executable.

If you have an odd number of nodes in the cluster, the subcluster that has more than half of the nodes available has Quorum. For example, on a three node cluster, the subcluster with two nodes available has Quorum. The other subcluster having only one node available gets neither operational nor configuration quorum and

therefore no resources will be started on that node. If a critical resource is already running on that node, the protection method defined in the ″CritRsrcProtMethod″ attribute will be applied to the node (see on page 138).

If you have an even number of nodes in the cluster and one of the subclusters consists of half of the nodes in case of a cluster split, then a tie breaker decides which subcluster is allowed to run critical resources. Nodes with critical resources that lose the competition for the tie breaker are subject to resource protection, which means that they will be stopped or rebooted immediately.

To perform an automatic tie breaking without operator action, you need a disk tie breaker (SCSI for Linux on System x, System p, and System i, or ECKD for Linux on System z, or DISK for AIX). A disk tie breaker is a shared disk which is accessible from all cluster nodes.

**SLES/10 x/Linux systems:** During startup, RSCT removes watchdog modules like `i8xx_tco` and `i6300esb`, which may be preloaded in order to be able to load the required softdog module. This means that you cannot have a hardware-related watchdog running in parallel to System Automation for Multiplatforms.

## VMTIMEBOMB function

The vmtimebomb function which is running for Linux on System z under VM ensures data protection for scenarios where z/VM gets held, but guest Linux systems keep on running. It is a new implementation of the protection methods described in the preceding section. Seen from the perspective of Topology Services, the vmtimebomb function is only indirectly accessed through a special VM vmwatchdog module. This module is similar to the ubiquitous Linux softdog watchdog module.

Watchdog modules are intended to prevent a permanent failure by automatically restarting the operating system during some sort of catastrophic failure. Typically, an application actively and regularly 'pings' the watchdog (usually by writing to a special device) to indicate that it is alive. This serves as an indication of the health of the system as a whole. Should the application fail to indicate it is alive, this indicates that there is a serious malfunction.

Something must monitor the status of the watchdog to take action when the application ceases to give the ″all's well″ report. With softdog, this is just the Linux operating system. In particularly dire circumstances, the operating system may fail and not be able to respond. With the vmwatchdog, external supervision is handled by the z/VM Control Program, which can restart the operating system on the virtual machine even in the worst cases. The vmwatchdog is only supported for 2.6 kernel versions running as guest systems under z/VM 5.1.0 (or later).

## Setting critical resources

Use the ProtectionMode persistent attribute to specify whether the resource is critical. If it is critical, then the Configuration RM (IBM.ConfigRM) decides whether the resource can be started as requested. The attribute may have the integer values 0 (non–critical) or 1 (critical). Per default IBM.Application resources are non–critical, and IBM.ServiceIP resources are critical. If the resource is set to critical, monitoring will immediately start.

Issue the RSCT command lsrsc to list the value of the ProtectionMode attribute:

```
lsrsrc IBM.Application Name NodeNameList ProtectionMode
```

Issue the RSCT command chrsrc to define the resource as critical by setting the ProtectionMode to 1:

```
chrsrc -s "Name='apache1'" IBM.Application ProtectionMode=1
```

To define a resource as non–critical set ProtectionMode to 0:

```
chrsrc -s "Name='apache1'" IBM.Application ProtectionMode=0
```

Issue the following to verify whether critical resources are currently active on a node for resource class IBM.Application:

```
lsrsrc IBM.Application Name NodeNameList OpState ProtectionMode
```

This provides the following output:

```
resource 1:
        Name           = "apache1"
        NodeNameList   = {"node1","node2"}
        OpState        = 1
        ProtectionMode = 1
resource 2:
        Name           = "apache1"
        NodeNameList   = {"node1"}
        OpState        = 2
        ProtectionMode = 1
resource 3:
        Name           = "apache1"
        NodeNameList   = {"node2"}
        OpState        = 1
        ProtectionMode = 1
```

Critical resource apache1 is active on the node2.

Issue the following to verify if critical resources are currently active on the nodes:

```
lsrsrc IBM.PeerNode Name CritRsrcActive
```

The output is as follows:

```
Resource Persistent and Dynamic Attributes for IBM.PeerNode
resource 1:
        Name           = "node1"
        CritRsrcActive = 0
resource 2:
        Name           = "node2"
        CritRsrcActive = 1
```

Critical resources are active on node2.

## Getting quorum information

Use the **lssrc** command for the IBM.RecoveryRM daemon to obtain the current quorum states.

```
node02:~/build # lssrc -ls IBM.RecoveryRM
```

You get the following output:

```
Daemon State:
   My Node Name         : node02
   Master Node Name     : node01 (node number = 1)
   Our CVN              : 61035379498
   Total Node Count     : 2
   Joined Member Count  : 2
   Config Quorum Count  : 2
   Startup Quorum Count : 1
   Operational Quorum State : HAS_QUORUM
   In Config Quorum     : TRUE
   In Config State      : TRUE
```

The meaning of the various attributes is as follows:

**Total Node count**
Is the number of nodes defined in the cluster.

**Joined Member Count**
is the number of IBM.RecoveryRM daemons running in the cluster. This is equivalent to the number of active nodes in the (sub)cluster.

**Config Quorum Count**
is the number of IBM.RecoveryRM daemons that must be active in order to make a configuration change by means of the IBM Tivoli System Automation commands.

**Startup Quorum Count**
Is the number of IBM.RecoveryRM daemons that must be active before the IBM Tivoli System Automation automation engine is activated.

**Operational Quorum State**
Indicates (sub)cluster wide whether this subcluster can survive or must immediately dissolve in case critical resources are running on the node(s) in the subcluster. The operational quorum state is provided by the dynamic attribute OpQuorumState of PeerDomain class. OpQuorumState can have the following values:

**0 – HAS_QUORUM**
IBM Tivoli System Automation may start resources

**1 – PENDING_QUORUM**
Indicates that a tie situation occurred that is not yet resolved. IBM Tivoli System Automation does not start resources.

**2 – NO_QUORUM**
IBM Tivoli System Automation is not allowed to start resources.

**In Config Quorum**
Indicates whether enough nodes hosting IBM.RecoveryRM daemons are active to accept configuration changes by IBM Tivoli System Automation commands. Shows TRUE if the total number of ″joined″ IBM.RecoveryRM daemon group members within the cluster is equal or above the Config Quorum count.

**In Config State**
Indicates whether the master IBM.RecoveryRM daemon has completed the verification of the system registry content at startup time. If the state equals to FALSE, any IBM Tivoli System Automation command will be rejected.

Enter the following to list OpQuorumState:

```
lsrsrc IBM.PeerDomain Name OpQuorumState
```

You get the following output:

```
Resource Persistent and Dynamic Attributes for:IBM.PeerDomain
resource 1:
        Name = "myCluster"
        OpQuorumState = 0
```

## Setting up and administering a tie breaker

The IBM.TieBreaker resource class allows you to configure a tie breaker such as ECKD or SCSI. Additionally two tie breakers are predefined, Operator and Fail. The operator tie breaker provides an undetermined result when a tie occurs and it is left to the administrator to resolve the tie through granting or denying the operational quorum. When a tie occurs and a tie breaker of type ″Fail″ is active, the attempt to reserve the tie breaker is always denied. Default tie breaker type is set to 'Operator'.

To list the available tie breaker type:

```
lsrsrc -c IBM.TieBreaker
```

You get the following output on a Linux system running on System x, System p, or System i:

```
Resource Class Persistent Attributes for: IBM.TieBreaker
        resource 1:
                 AvailableTypes ={["SCSI",""],["EXEC",""],["Operator",""],["Fail",""]}
```

To list the tie breaker name:

```
lsrsrc IBM.TieBreaker
```

You get the following output:

```
Resource Persistent Attributes for: IBM.TieBreaker
        resource 1:
            Name                = "FAIL"
            Type                = "FAIL"
            DeviceInfo          = ""
```

```
            ReprobeData         = ""
            ReleaseRetryPeriod  = 0
            HeartbeatPeriod     = 0
            PreReserveWaitTime  = 0
            PostReserveWaitTime = 0
            NodeInfo            = {}

        resource 2:
            Name                = "Operator"
            Type                = "Operator"
            DeviceInfo          = ""
            ReprobeData         = ""
            ReleaseRetryPeriod  = 0
            HeartbeatPeriod     = 0
            PreReserveWaitTime  = 0
            PostReserveWaitTime = 0
            NodeInfo            = {}

resource 3:
           Name                 = "myTieBreaker"
           Type                 = "SCSI"
           DeviceInfo           = "ID=0 LUN=0 CHAN=0 HOST=2"
           ReprobeData          = ""
           ReleaseRetryPeriod   = 0
           HeartbeatPeriod      = 5
           PreReserveWaitTime   = 0
           PostReserveWaitTime  = 0
           NodeInfo             = {}

resource 4:
           Name                 = "mytb"
           Type                 = "EXEC"
           DeviceInfo           = "PATHNAME=/usr/sbin/rsct/bin/samtb_net
                                           Address=192.168.177.2"
           ReprobeData          = ""
           ReleaseRetryPeriod   = 0
           HeartbeatPeriod      = 30
           PreReserveWaitTime   = 0
           PostReserveWaitTime  = 30
           NodeInfo             = {}
           ActivePeerDomain     = "21"
```

Although you can define several tie breaker resources in the resource class IBM.TieBreaker, only one of them can be active in the cluster at the same time. Issue the following command to list the tie breaker that is currently active in the cluster:

```
lsrsrc -c IBM.PeerNode OpQuorumTieBreaker
```

You get the following output:

```
Resource Class Persistent Attributes for: IBM.PeerNode
        resource 1:
          OpQuorumTieBreaker = "Operator"
```

The active tie breaker is set with the following command:

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

To grant/deny the operational quorum when tie breaker is ″Operator″:

```
runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=1 (0 to deny)
```

**Note:** In order to avoid race conditions, the operator tie breaker must be denied first for the losing subcluster(s) before granting it to the subcluster which is supposed to continue.

## Using a tie breaker

To create a basic setup for the tie breaker you need a cluster of two (or other even number) of nodes. Also you need a disk that is shared between all nodes of the cluster. The tie breaker disk is shared between all cluster nodes.

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

The following three examples show how to use a tie breaker with an ECKD, SCSI or DISK device. Note that the tie breaker needs not be formatted or partitioned. Then it will only be marked active without size information (in case of ECKD).

### Example 1: ECKD tie breaker setup for a two node cluster

Note the following when defining the tie breaker disk under VM:

- Full pack minidisk should be defined.
- If minidisk cache is used, its value should be set to off.
- ECKD disk is shared between both nodes.

The ECKD tie breaker type is specific for Linux on System z. If you want to create an ECKD tie breaker object, you need to set the DeviceInfo persistent resource attribute to indicate the ECKD device number. This type of tie breaker uses a reserve/release mechanism and needs to be re-reserved periodically to hold the reservation. For this reason, we strongly recommend that you also specify the HeartbeatPeriod persistent resource attribute when creating a tie breaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation request is re-issued.

Collect the following system information (Linux kernel 2.4)::

```
node01:~ # cat /proc/subchannels
Device sch.  Dev Type/Model CU  in use  PIM PAM POM CHPIDs
------------------------------------------------------------------------
50DE   0A6F  3390/0A   3990/E9          F0  A0  FF  7475E6E7 FFFFFFFF
node01:~ # cat /proc/dasd/devices
50dc(ECKD) at ( 94:  0) is           : active at blocksize: 4096, 601020 blocks, 2347 MB
50dd(ECKD) at ( 94:  4) is           : active at blocksize: 4096, 601020 blocks, 2347 MB
50de(ECKD) at ( 94:  8) is           : active at blocksize: 4096, 601020 blocks, 2347 MB
50df(ECKD) at ( 94: 12) is           : active at blocksize: 4096, 601020 blocks, 2347 MB
```

For Linux kernel 2.6 use the *lscss* command instead of the *cat /proc/subchannels* command.
Perform the following steps to use the tie breaker:

1. Create a tie breaker resource object in IBM.TieBreaker class. **DeviceInfo** shows the ECKD device number. It can be obtained from /proc/dasd/devices file.

```
node01:~ # mkrsrc IBM.TieBreaker Name=myTieBreaker Type=ECKD DeviceInfo="ID=50de" HeartbeatPeriod=5
node01:~ # lsrsrc IBM.TieBreaker
Resource Persistent Attributes for: IBM.TieBreaker
resource 1:
        Name                = "Operator"
        Type                = "Operator"
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
resource 2:
        Name                = "Fail"
        Type                = "Fail"
```

```
        DeviceInfo          = ""
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 0
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
resource 3:
        Name                = "myTieBreaker"
        Type                = "ECKD"
        DeviceInfo          = "ID=50de"
        ReprobeData         = ""
        ReleaseRetryPeriod  = 0
        HeartbeatPeriod     = 5
        PreReserveWaitTime  = 0
        PostReserveWaitTime = 0
        NodeInfo            = {}
```

2. Change OpQuorumTieBreaker attribute in IBM.PeerNode class to one of the tie breaker resource objects.

```
node01:~ # chrsrc -c IBM.PeerNode OpQuorumTieBreaker="myTieBreaker"
```

```
node01:~ # lsrsrc -c IBM.PeerNode
Resource Class Persistent Attributes for: IBM.PeerNode
resource 1:
        CommittedRSCTVersion  = ""
        ActiveVersionChanging = 0
        OpQuorumOverride      = 0
        CritRsrcProtMethod    = 1
        OpQuorumTieBreaker    = "myTieBreaker"
```

**Hint:** If the node reserving a tie breaker is down and cannot be rebooted, manual access on another node is needed to break the reservation and take it over on the other node. The tie breaker disk can

- be either still attached to the healthy node, provided this node has not been rebooted in the mean time:

```
node01:~ # cat /proc/subchannels
Device sch. Dev Type/Model CU in use PIM PAM POM CHPIDs
------------------------------------------------------------
50DE 0A6F 3390/0A 3990/E9 F0 A0 FF 7475E6E7 FFFFFFFF

node01:~ # cat /proc/dasd/devices
50de(ECKD) at ( 94: 8) is dasdc: active at blocksize: 4096,601020 blocks, 2347 MB
```

- be boxed, if this node has been rebooted and cannot recognize the tie breaker disk anymore:

```
node01:~ # cat /proc/subchannels
Device sch. Dev Type/Model CU in use PIM PAM POM CHPIDs
------------------------------------------------------------
50DE   0A6F            FFFF/00          F0  A0  FF  7475E6E7 FFFFFFFF

node01:~ # cat /proc/dasd/devices
50de(ECKD) at ( 94: 8) is dasdc : boxed
```

  To break the tie breaker disk reservation enter the command /usr/sbin/rsct/bin/tb_break:

```
tb_break -t ECKD /dev/dasdc
```

  The tie breaker disk should now be reserved by the healthy node.

**Note:** If the tb_brk command does not work the first time, issue it again.

## Example 2: SCSI tie breaker setup for a two node cluster

The SCSI tie breaker type is specific for Linux on System x, pSeries, and iSeries. If you want to create an SCSI tie breaker object, you need to specify the SCSI device using the DeviceInfo persistent resource attribute. If the SCSI configuration is different on different nodes in the cluster, you can also use the NodeInfo persistent resource attribute to reflect those differences. This type of tie breaker uses a reserve/release mechanism and needs to be re-reserved periodically to hold the reservation. For this reason, we strongly recommend that you also specify the HeartbeatPeriod persistent resource attribute when creating a tie breaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation request is re-issued.

SCSI devices can be identified by four integer values for the attributes HOST, CHAN, ID, and LUN:

```
node1:~# dmesg | grep "Attached scsi disk"
```

Normally these parameters are identical on each cluster node.
For example,for node1 and node2 these are HOST=0 CHAN=0 ID=4 LUN=0.
You can then create the tie breaker object:

```
mkrsrc IBM.TieBreaker Name=myTieBreaker Type=SCSI DeviceInfo=" HOST=0 CHAN=0 ID=4 LUN=0"
```

The four values above may also be different for different nodes (even if the target device is same). In that case the **NodeInfo** field should be used.
Use the four integer values from the command output:

```
# dmesg | grep "Attached scsi disk"
Attached scsi disk sdf at scsi2, channel 2, id 4, lun 0
```

For disk sdf is HOST=2, CHAN=2, ID=4, LUN=0.
For example, a SCSI device is connected to 2 nodes named node1 and node2 and has the following SCSI identifiers:

```
node1:  HOST=0 CHAN=0 ID=4 LUN=0
node2:  HOST=2 CHAN=2 ID=4 LUN=0
```

You can then create the tie breaker object as

```
# mkrsrc IBM.TieBreaker Name=scsi Type=SCSI DeviceInfo="ID=4 LUN=0"
NodeInfo='{["node1", "HOST=0 CHAN=0"], ["node2", "HOST=2 CHAN=2"]}'
```

IBM Tivoli System Automation handles **DeviceInfo** and **NodeInfo** in such a way that it merges the two strings, **DeviceInfo** first and then **NodeInfo**. For example, for node1 the merged string is

```
"ID=4 LUN=0 HOST=0 CHAN=0"
```

which will then be parsed.

Also, any duplicated keywords will be allowed and the last one will be used. Therefore, the same command can be specified as

```
# mkrsrc IBM.TieBreaker Name=myTieBreaker Type=SCSI DeviceInfo="ID=4 LUN=0
HOST=0,CHAN=0"  NodeInfo='{["node2", "HOST=2 CHAN=2"]}'
```

This simplification may be useful as often and most likely the SCSI id is the same for many nodes.

**Hint:** If the node reserving a tie breaker is down and cannot be rebooted, manual access on another node is needed to release the SCSI tie breaker disk. To release a disk, run the command:

```
tb_break [–f] HOST CHAN ID LUN
```

for example,

```
/usr/sbin/rsct/bin/tb_break –f HOST=0 CHAN=0 ID=4 LUN=0
```

### Example 3: AIX DISK tie breaker setup for a two node cluster

The DISK tie breaker type is specific to AIX. If you want to create a DISK tie breaker object, you need to set the DeviceInfo persistent resource attribute to indicate the AIX device name. The AIX device name must specify a SCSI or SCSI-like physical disk that is shared by all nodes of the peer domain.

Physical disks attached via Fiber Channel, iSCSI, and Serial Storage Architecture may serve as a DISK tie breaker. However, IDE hard disks do not support the SCSI protocol and cannot serve as a DISK tie breaker. Logical volumes also cannot serve as a DISK tie breaker. This type of tie breaker uses a reserve/release mechanism and needs to be re-reserved periodically to hold the reservation. For this reason, we strongly recommend that you also specify the HeartbeatPeriod persistent resource attribute when creating a tie breaker of this type. The HeartbeatPeriod persistent resource attribute defines the interval at which the reservation request is re-issued.

To print every known physical volume in the system along with its physical disk name, enter the **lspv** command:

```
lspv
```

An output similiar to the following one is displayed:

```
hdisk0 000000371e5766b8 rootvg active
hdisk1 000069683404ed54 None
```

In order to verify that a disk is a SCSI or SCSI-like disk and so a suitable candidate for a DISK tie breaker, use the **lsdev** command. For example:

```
lsdev -C -l hdisk1
```

An output similiar to the following one is displayed:

```
hdisk1 Available 10-60-00-0,0 16 Bit SCSI Disk Drive
```

In order to serve as a tie breaker disk, the disk must be shared by all nodes of the peer domain. Check the physical volume ID returned by the lspv command to determine if the disk is shared between nodes (in the preceding output for the lspv command, the physical volume ID is listed in the second column; the volume ID for hdisk1 is 000069683404ed54.) Be aware, however, that AIX remembers all disks that have been attached to the system, and the disks listed by the lspv command may no longer be attached. If such a disk was moved to another machine, it might appear as if the disk is shared, when in fact it is no longer attached to the original machine.

The disk on which IBM.TieBreaker resources are stored should not also be used to store file systems. If the nodes of the cluster share more than one disk, it may be difficult to determine which one is the tie breaker disk, and which one is used for application data. The output from the lsdev command shows the SCSI address associated with the disk. (In the preceding output for the lsdev command, the SCSI address is listed in the third column; the SCSI address for hdisk0 is 10-60-00-0,0.) This information will help you to identify the correct disk if you are aware of the address of the disk prior to its installation.

Once you know the device name, you can issue the **mkrsrc** command:

```
mkrsrc IBM.TieBreaker Name=myTieBreaker Type=DISK DeviceInfo="DEVICE=/dev/hdisk1" HeartbeatPeriod=5
```

**Hint:** The tie breaker relies on SCSI-2 reservation, which is not necessarily supported by every combination of storage and driver setup. In order to check that the setup supports SCSI-2 reservation, RSCT comes with the disk_reserve utility that must be invoked with its full path (/usr/sbin/rsct/bin/disk_reserve).

The tie breaker will work correctly if the tie breaker disk can be reserved and unlocked from either node and if the disk cannot be reserved while it is locked by the other node.

**Synopsis:**
**Usage:**

```
/usr/sbin/rsct/bin/disk_reserve [-l | -u | -b] [-h] [-v] [-f] [-d sdisk_name]
/usr/sbin/rsct/bin/disk_reserve [-l | -u | -b] [-h] [-v] [-f] [-g sg_device_name]
```

**-h** - display this help text

**-v** - verbose

**-f** - reserve after break (for the **-l** or **-b** option)

**-d sdisk_name** - disk to operate (for example, /dev/sdb)

**-l** - lock (reserve)

**-u** - unlock (release)

**-b** - break

**-g sg_device_name** (for example, /dev/sg1)

**Examples:**

```
/usr/sbin/rsct/bin/disk_reserve -l -f -d /dev/sde
/usr/sbin/rsct/bin/disk_reserve -l -g /dev/sg3
```

If the node reserving a tie breaker is down and cannot be rebooted, manual access on another node is needed to release the SCSI tie breaker disk. To release the disk, run the command:

```
/usr/sbin/rsct/bin/tb_break –f –t DISK "DEVICE=/dev/hdisk1"
```

Execute the **lspath** command, for example:

```
lspath -l hdisk2
lspath: 0514-538 Cannot perform the requested function because the
                 specified device does not support multiple paths.
```

**Sample output:**
In contrast to the sample above, this sample shows an example that does not support SCSI-2 reservation and hence not the tie breaker.

```
#lspath -l hdisk2
Enabled hdisk2 fscsi0
Failed  hdisk2 fscsi0
Failed  hdisk2 fscsi0
Failed  hdisk2 fscsi0
Failed  hdisk2 fscsi0
Enabled hdisk2 fscsi0
Enabled hdisk2 fscsi0
Enabled hdisk2 fscsi1
```

```
Failed  hdisk2 fscsi1
Failed  hdisk2 fscsi1
Failed  hdisk2 fscsi1
Failed  hdisk2 fscsi1
Enabled hdisk2 fscsi1
Enabled hdisk2 fscsi1
```

## Network tie breaker

The network tie breaker provides an alternative to the disk and operator based tie breakers described in the preceding sections of this chapter. It uses an external IP (network instance) to resolve a tie situation.

There may be several reasons to use a network tie breaker, for example:

- There is no possibility to use a disk based tie breaker.
- It is the highest priority of a high availability environment to communicate with instances outside the cluster.

Example: The primary functions of a web server are to deliver web pages to clients outside of the cluster. In order to make this service highly available, the tie breaker should not grant access to a node which is not able to communicate to instances outside of the cluster.

**Requirements for the network tie breaker**

To ensure the network tie breaker functionality the external IP instance must be reachable from all nodes within the highly available cluster. Also the external IP instance must be able to reply to ICMP echo requests (ping). If you install a firewall rule which will blocks ICMP traffic between the cluster nodes and the external IP instance, the network tie breaker will not work. The biggest danger is that the cluster nodes cannot communicate to their peers (cluster split), but both sub clusters are able to reach the external IP instance. Under normal conditions IP ensures the following: If both sub clusters can reach the external gateway, they are also able to communicate to their peers. There might be unusual IP setups which will not ensure this rule (e.g. firewall settings). If you cannot ensure this rule, you cannot use the network tie breaker.

The following table shows the advantages and disadvantages of the two types of tie breakers:

*Table 19. Comparison of network based tie breaker and disk based tie breaker*

| Network based tie breaker | Disk based tie breaker |
|---|---|
| + No hardware dependency<br>+ Evaluates availability of communication | + Most secure tie breaker. Hardware takes care that only one instance (node) is able to get the tie breaker. |
| - If the external IP instance is not available in case of a cluster split no subcluster will get quorum.<br>- There may be error conditions in which a tie situation occurs, but more than one node is able to communicate. In this case there is a slight possibility both subclusters are able to get the tie breaker. | - In case of a loss in communication this tie breaker may grant access to a node which is not able to communicate to instances outside of the cluster. |

## Setting up a network tie breaker

The network tie breaker is realized as an RSCT exec tie breaker. See the RSCT documentation if you want to learn more about an exec tie breaker- The network tie breaker executable *samtb_net* is located in the **/usr/sbin/rsct/bin** directory. The current implementation knows following options which have to be specified as key value pairs during creation of the RSCT exec tie breaker:

Address=<IP address>
: Address of the external IP instance which should be used to resolve the tie situation. Specify an IPv4 address in dotted quote notation, for example 192.168.1.1. Do not use a DNS name, because in case of a communication problem, which may typically occur during cluster split, DNS may not work properly. Address is a mandatory option, no default is present.

Log=<1/0>
: Specify 1 if you want the network tie breaker to write logs to the system log facility (syslog). Default is 1. Allowed values are 1 and 0.

Count=<number>
: Numbers of ICMP echo requests which are sent to determinate quorum. If the first request gets a response no further requests are sent. Default is 2. Allowed value range is from 1 to 9.

The following command will create a new network tie breaker:

```
# mkrsrc IBM.TieBreaker Type="EXEC" Name="mynetworktb"
DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=192.168.1.1
Log=1' PostReserveWaitTime=30;
```

Activate your network tie breaker as follows:

```
# chrsrc -c IBM.PeerNode OpQuorumTieBreaker="mynetworktb"
```

You can use any regular RSCT command to manipulate the network tie breaker definition. Use the ″rmrsrc″ command to delete the tie breaker definition.

**Some background information about the network tie breaker:** The RSCT tie breaker design is based on the idea that a node takes something away (reserve), and that therefore after that another node is not able to also take it because it is not available anymore.

As this is not possible for a network based approach (network tie breaker), some restrictions were introduced to the base tie breaker design.

Reserve behavior: After there was an unsuccessful reserve attempt, another reserve is not allowed until the node has joined the cluster again. To ensure that a file is written to /var/ct/ which indicates that there was a failed reserve before. If this file is present, a call to tie breaker reserve will always fail. There is an additional process forked which watches quorum and removes the block file if the node has joined the domain again.

The following sample file was created by the network tie breaker as the result of a failing tie breaker reserve operation to the external IP instance 192.168.1.1. It contains the timestamp of the failed reserve operation.

```
# cat /var/ct/samtb_net_blockreserve_192.168.1.1
Mo Jul 4 08:38:40 CEST 2005
```

**Configuring a RSCT tie breaker resource for the network tie-breaker:** The following explains the most important configuration options of a RSCT tie breaker definition. This gives an idea how they can be configured for the network tie breaker.

**PostReserveWaitTime=30**

In case of a failing reserve ConfigRM will periodically call tie breaker reserve operation. Since the network tie-breaker only honors the first reserve attempt and blocks periodical reserve in case there was a failed reserve before, the PostReserveWaitTime should be set to the maximum possible value which is 30 seconds. This will keep the system load low in case a node stays in the pending Quorum state (periodical calls to tie breaker reserve).

**HeartbeatPeriod=30**

After there was a successful reserve, ConfigRM starts calling periodical tie breaker heartbeat operation. To keep the system load low during a cluster split, increase the time between the tie breaker heartbeats or even turn off heart beating by setting HeartbeatPeriod to 0.

**Reviewing the system logs of a network tie breaker scenario:** Here are the system logs of a two node cluster (node n1 and node n2). For the error scenario it is assumed there are no critical resources running on both nodes. A network problem will break all available communication paths between the peers, but one peer (n2) is still able to communicate to its gateway (192.168.177.2). After some time communication is established again and both nodes can join the cluster.

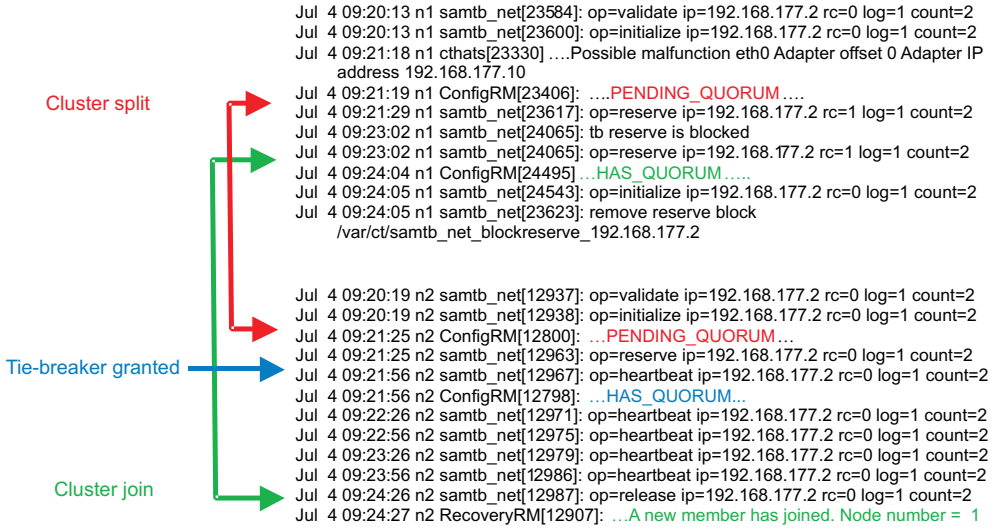



*Figure 10. Two node cluster system logs*

## Overriding the operational quorum

In order to remove nodes from the cluster, at least one node of the cluster must be online to initiate the **rmrpnode** command. If there are not enough nodes to ever achieve an operational quorum, there is no chance to adjust the cluster size by administrative means so that the quorum can be reestablished.

If for any reasons the operational quorum function must be deactivated, the persistent attribute OpQuorumOverride must be set to 1:

```
chrsrc –c IBM.PeerNode OpQuorumOverride=1
```

In this case operational quorum State is always HAS_QUORUM and resource protection is not ensured anymore

# Chapter 10. Setting up a highly available network

When talking about setting up a highly available network, we should distinguish between two situations:

- A more reliable cluster communication in which the cluster infrastructure (RSCT) takes care that all available communication paths are used to ensure cluster integrity and configuration data replication.
- A representation of a highly available IT service in which automation takes care of an IP address (further called ServiceIP) which represents an IT service to clients outside the cluster.

The same communication interface is often used for both tasks, but this is not necessary and often not the best way to do. The following section describes this.

## Considerations when planning a highly available network setup

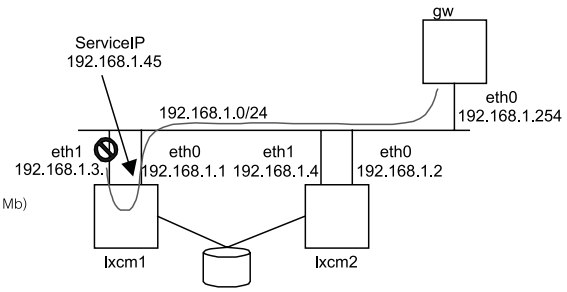
This section is intended to help you to understand the complexity of a highly available network and provides some questions which can help you to plan the setup of a highly available network.

### What makes high availability of a network infrastructure difficult?

Ensuring high availability of the network infrastructure is not as easy as plugging in another network interface in an existing node. Of course you can plug in the new device and configure it with an address from the existing network, but this will not work in case the "wrong" interface dies. Here is an example from a Linux setup:

```
eth0   Link encap:Ethernet  HWaddr 00:00:00:00:00:00
       inet addr:192.168.1.1  Mask:255.255.255.0
       inet6 addr: fe80::200:ff:fe00:0/10 Scope:Link
       UP RUNNING NOARP MULTICAST  MTU:1492  Metric:1
       RX packets:1147264 errors:0 dropped:0 overruns:0 frame:0
       TX packets:1557235 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       RX bytes:873548285 (833.0 Mb)  TX bytes:674939696 (643.6 Mb)
       Interrupt:2

eth1   Link encap:Ethernet  HWaddr 00:00:00:00:00:00
       inet addr:192.168.1.3  Mask:255.255.255.0
       inet6 addr: fe80::200:ff:fe00:0/10 Scope:Link
       UP RUNNING NOARP MULTICAST  MTU:1492  Metric:1
       RX packets:297057 errors:0 dropped:0 overruns:0 frame:0
       TX packets:289815 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       RX bytes:30153527 (28.7 Mb)  TX bytes:38726923 (36.9 Mb)
       Interrupt:5
```




```
Kernel IP routing table
Destination Gateway     Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *       255.255.255.0   U     0      0        0 eth1
192,168.1.0     *       255.255.255.0   U     0      0        0 eth0
default     192.168.1.254 0.0.0.0       UG    0      0        0 eth1
```

*Figure 11. Problems planning a highly available network*

Every static configured network device causes an entry in the routing table. The routing algorithm chooses the first matching route out of this table. In this

example, device eth1 on node lnxcm1 fails. As eth1 is the first entry in the routing table, the node cannot send packages out to the network although there is another working network interface (eth0).

This chapter gives you a couple of suggestions how you can make a more sophisticated approach to ensure the high availability of your network infrastructure. Like in many other cases the most complicate approach (dynamic routing) will be the best, but you may want to consider the second or third best approach to keep complexity and implementation effort low.

### Things to clarify before planning a high availability network

Clarify the following questions before you start planning your high availability network:

1. What kind of a high availability network do you need? Is it necessary to move a ServiceIP from one interface to another on the same node, or is it also appropriate to switch to another node which has a working interface in the required subnet?
2. Can you implement additional IP subnets or do you have to use an existing network infrastructure?
3. Do you only work in the scope of our cluster nodes or are you able to implement/deploy network services on other nodes outside of the automation cluster?
4. What kind of network hardware do you have?
5. How much effort do you want to invest?
6. How much complexity do you want to introduce?
7. What skills do we have?

Depending on the answers to the questions above you may want to choose one of the setups described in this chapter to develop your own high availability network strategy.

## Running a one or two node cluster: detecting network interface failures

In case you run a one or two node cluster you need some additional configuration to detect network interface failures. The cluster software periodically tries to reach each network interface of the cluster. If there is a two node cluster and one interface fails on one node, the other interface on the other node is not able to get response from the peer and will also be flagged offline.

To avoid this behavior the cluster software must be told to contact a network instance outside the cluster. Best practise is to use the default gateway of the subnet the interface is in.

On each node create following file:

/usr/sbin/cluster/netmon.cf

Each line of this file should contain the machine name or IP address of the external instance. An IP address should be specified in dotted decimal format.

This is an example of a `/usr/sbin/cluster/netmon.cf` file:

```
# this is default gateway for all interfaces in 192.168.1.0 network
192.168.1.1

# this is default gateway for all interfaces in 192.168.2.0 network
gw.de.ibm.com
```

## Two node cluster, each node has one ethernet interface

The following network setup is given:

| | Name: | Device: | IP: |
|---|---|---|---|
| Cluster node | lnxcm1 | eth0 | 9.152.172.1/24 |
| Cluster node | lnxcm2 | eth0 | 9.152.172.2/24 |
| Router | gw | eth0 | 9.152.172.254/24 |
| ServiceIP | – | – | 9.152.172.3/24 |

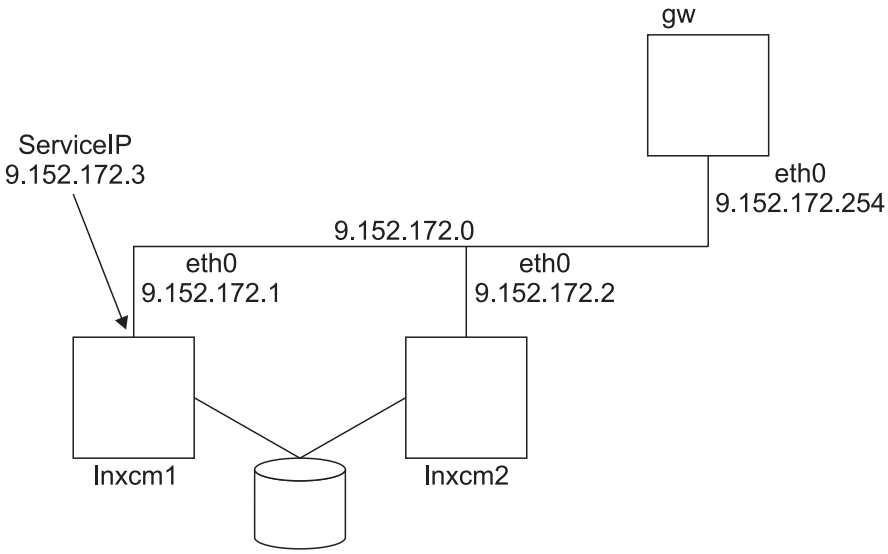



*Figure 12. Two nodes, one interface*

In this setup the cluster communication and the presentation of the highly available IT service uses the same communication path, the 9.152.172.0 network.

Automation can assign the ServiceIP either on the lnxcm1 interface eth0 or on the lnxcm2 interface eth0. If one interface fails. automation moves the ServiceIP to the other node. Thus it satisfies the policy which requires assigning the ServiceIP on a running network interface.

In this setup the failure of one network interface will lead to a break in the cluster communication with all the problems as described in Chapter 9, “Protecting your resources – quorum support,” on page 137. If the communication breaks as shown in Figure 13 on page 159, the tie breaker decides which node is able to go on with automation. If this is node lnxcm1, automation will find on lnxcm1 no online network interface to assign the ServiceIP on.

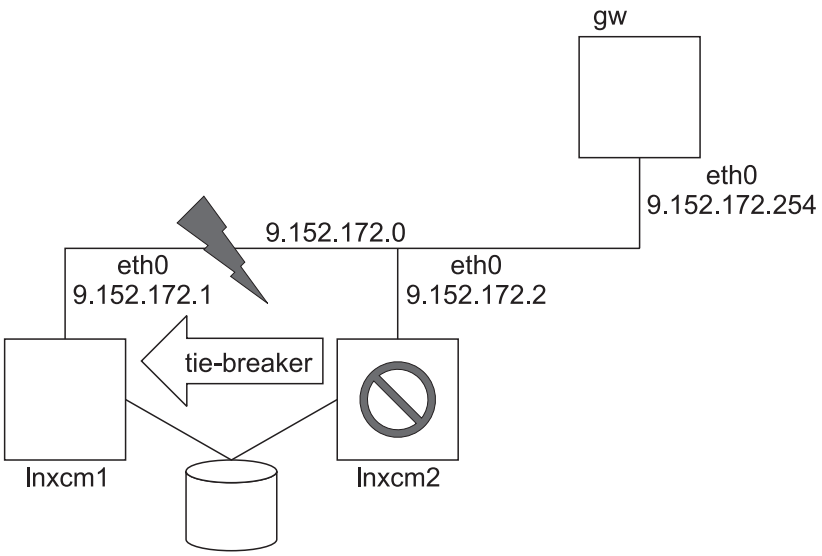



*Figure 13. Two nodes, one interface – interface failure*

In this example the network 9.152.172.0 served two purposes:

1. Representing the network for the highly available IT service.
2. Used for internal cluster communication.

Sample IBM Tivoli System Automation policy:

```
lnxcm1# mkequ NetInt IBM.NetworkInterface:eth0:lnxcm1,eth0:lnxcm2
lnxcm1# mkrsrc IBM.ServiceIP Name="SIP"
IPAddress="9.152.172.3"
NetMask="255.255.255.0"
NodeNameList="{'lnxcm1','lnxcm2'}"
lnxcm1# mkrg rg
lnxcm1# addrgmbr -g rg IBM.ServiceIP:SIP
lnxcm1# mkrel -p dependson -S IBM.ServiceIP:SIP -G IBM.Equivalency:NetInt
```

| Advantage | Disadvantage |
|---|---|
| Very easy setup. | Each communication problem leads to cluster split. |
| Less network hardware required. | ServiceIP moves only between nodes. |

## Two node cluster, each node has two network interfaces

Before starting with this setup keep in mind that it is not possible to have more than one static configured network interface in the same IP subnet. Each IP address will cause an entry in the kernel routing table. In case of two interfaces in the same subnet there will be 2 routes for the same subnet. If the interface, which created the first entry, fails the communication for this subnet will break down even if there is another interface which still is able to communicate.

## Two physically separated networks, move ServiceIP between nodes

The following network setup applies:

| | **Name:** | **Device:** | **IP:** |
|---|---|---|---|
| Cluster node | lnxcm1 | eth0<br>eth1 | 9.152.172.1/24<br>192.168.1.1/24 |
| Cluster node | lnxcm2 | eth0<br>eth1 | 9.152.172.2/24<br>192.168.1.2/24 |
| Router | gw | eth0 | 9.152.172.254/24 |
| ServiceIP | - | – | 9.152.172.3/24 |

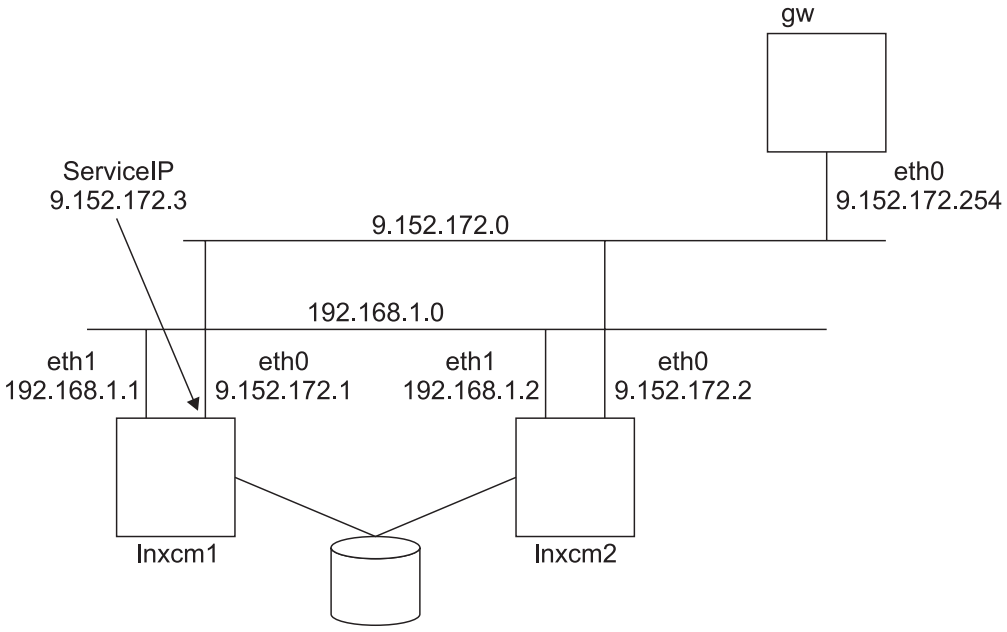



*Figure 14. Two nodes, two interfaces, two physically separated networks*

There are now two networks 192.168.1.0 and 9.152.172.0 for the cluster communication. If there is a failure in one network interface the cluster will not break.

- Network 9.152.172.0 represents the network for the highly available IT service.
- Network 192.168.1.0 makes cluster internal communication more reliable.

Since only the network of the ServiceIP is connected to the gateway, a failure of interface eth0 on lnxcm1 will cause the automation to move the ServiceIP to the

interface eth0 on the other node lnxcm2. Because of the physical separation of the two networks it is not possible to move the ServiceIP from eth0 to eth1 within the same node.

The sample IBM Tivoli System Automation policy is the same as shown on page on page 159.

| Advantage | Disadvantage |
|---|---|
| Easy setup. | ServiceIP moves only between nodes. |
| Redundancy in cluster communication. | |

## Three logical networks in one physical network, move ServiceIP between network interfaces

Another network setup is required to not only move the ServiceIP between nodes in the cluster but also between interfaces within one node. Needed is a separate logical network for each interface of a node, and an additional network for the ServiceIP. Choosing an existing network (one of eth0 or eth1) would cause routing problems. Make sure to connect all interfaces to the same physical network. This allows each interface to hold addresses of all the logical networks.

The following network setup applies:

| | Name: | Device: | IP: |
|---|---|---|---|
| Cluster node | lnxcm1 | eth0<br>eth1 | 192.168.1.1/24<br>192.168.2.1/24 |
| Cluster node | lnxcm2 | eth0<br>eth1 | 192.168.1.2/24<br>192.168.2.2/24 |
| Router | gw | eth0 | 9.152.172.254/24 |
| ServiceIP | - | – | 9.152.172.3/24 |

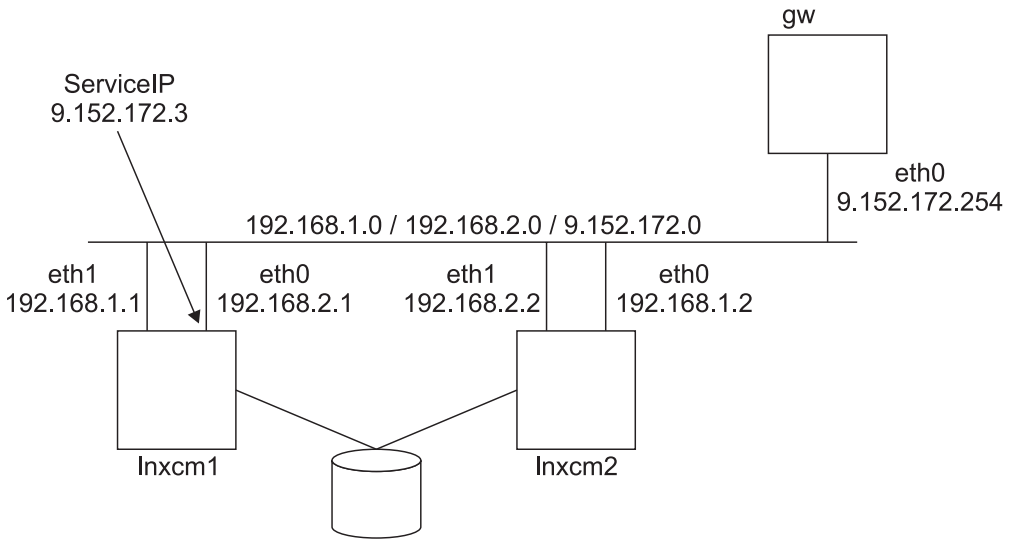



*Figure 15. Two nodes, two interfaces, one physical network*

- Network 9.152.172.0 represents the network for the highly available IT service.
- Network 192.168.1.0 represents the first cluster internal communication network.

- Network 192.168.2.0 represents the second cluster internal communication network.

Sample IBM Tivoli System Automation policy:

```
lnxcm1# mkequ NetInt
IBM.NetworkInterface:eth0:lnxcm1,eth1:lnxcm1,eth0:lnxcm2,eth1:lnxcm2
lnxcm1# mkrsrc IBM.ServiceIP Name="SIP" IPAddress="9.152.172.3"
NetMask="255.255.255.0" NodeNameList="{'lnxcm1','lnxcm2'}"
lnxcm1# mkrg rg
lnxcm1# addrgmbr -g rg IBM.ServiceIP:SIP
lnxcm1# mkrel -p dependson -S IBM.ServiceIP:SIP -G IBM.Equivalency:NetInt
```

| **Advantage** | **Disadvantage** |
|---|---|
| Easy setup. | 3 logical networks in 1 physical network. |
| Redundancy in cluster communication. | Traffic of 3 networks on 1 physical medium. |
| ServiceIP can move between interfaces and nodes | |

## Two physically separated networks, dynamic routing and VIPA

A detailed description of this setup is beyond the scope of this manual. Basically the ServiceIP is assigned to a virtual network within the kernel of a cluster node. Dynamic routing on all cluster nodes and the gateway makes sure that a route to the ServiceIP is established.

The following network setup applies:

| | **Name:** | **Device:** | **IP:** |
|---|---|---|---|
| Cluster node | lnxcm1 | eth0<br>eth1 | 9.152.170.1/24<br>9.152.171.1/24 |
| Cluster node | lnxcm2 | eth0<br>eth1 | 9.152.170.2/24<br>9.152.171.2/24 |
| Router | gw | eth0<br>eth1 | 9.152.170.254/24<br>9.152.171.254/24 |
| ServiceIP | - | – | 9.152.172.3/24 |

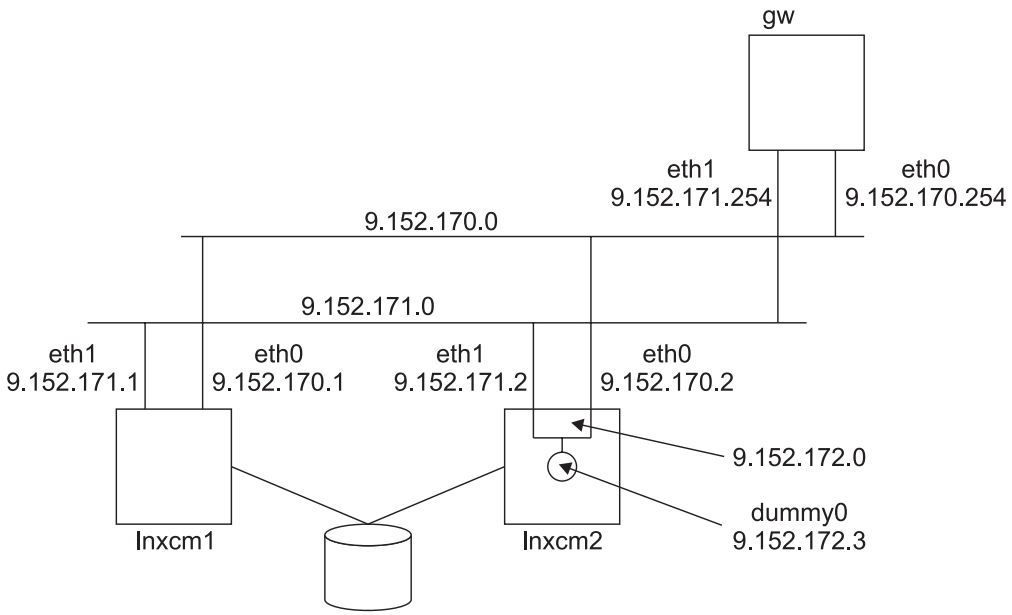



*Figure 16. Two physically separated networks, dynamic routing and VIPA*

| Advantage | Disadvantage |
|---|---|
| There is no dependency to the physical network device. | Complicate setup. |
| Concept of finding dynamically the best way to a host (IP address). | Dynamic routing required. |
| No need to move ServiceIP between interfaces | Setup is not restricted to the cluster nodes; gateway also has to support dynamic routing. |

## Interface bonding

Several physical network interfaces are bonded together to one logical network device. The operating system has to support this feature with a special bonding device driver. Consult your operating system documentation how to configure interface bonding on your system. Make sure that you configure HA (high availability) bonding and ensure your network interface cards support the interface failure detection mechanism your bonding driver requires.

The following network setup applies:

| | Name: | Device: | IP: |
|---|---|---|---|
| Cluster node | lnxcm1 | eth0<br>eth1 | 9.152.172.1/24<br>9.152.172.1/24 |
| Cluster node | lnxcm2 | eth0<br>eth1 | 9.152.172.2/24<br>9.152.172.2/24 |
| Router | gw | eth0 | 9.152.172.254/24 |
| ServiceIP | - | - | 9.152.172.3/24 |

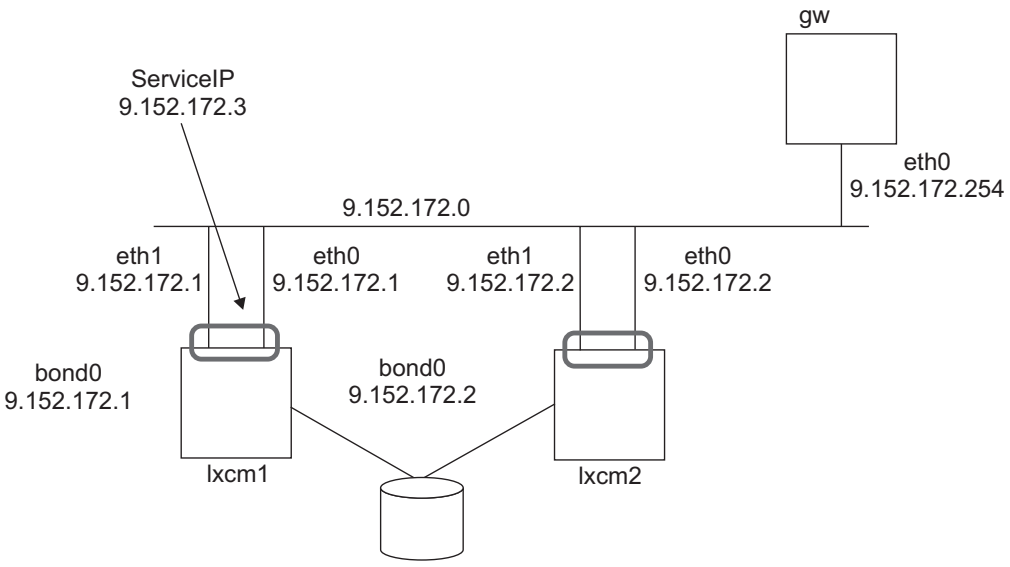



*Figure 17. Network interfaces bonded together to one logical network device*

| Advantage | Disadvantage |
|---|---|
| Easy setup. | Operating system has to support interface bonding. |
| Redundancy in cluster communication. | Network interface hardware may has to support interface failure detection (for example, MII link monitoring). |
| There is no need to move ServiceIP between devices on the same node. | |

# Chapter 11. Controlling and administering IBM Tivoli System Automation

This chapter describes various parameters which can be used to control and change the general behavior of IBM Tivoli System Automation. It also gives some insight into the infrastructure of IBM Tivoli System Automation and offers some helpful hints and tips.

## Controlling IBM Tivoli System Automation

There are several attributes you can use to change the general behavior of IBM Tivoli System Automation. You can start/stop the automation functionality, define some timeouts or exclude some nodes from automation, for example, for maintenance reasons.

These attributes are:

- TimeOut

  Specifies the timeout value in seconds for a start control operation executed by IBM Tivoli System Automation. After the timeout expires the operation is repeated if the RetryCount is not exceeded.
- RetryCount

  Number of allowed attempts if a control operation fails or times out.
- Automation

  Flag to enable or disable automation by IBM Tivoli System Automation.
- ExcludedNodes

  List of nodes on which IBM Tivoli System Automation actively pushes resources away or stops them. For example, this can be used for maintenance reasons.
- ResourceRestartTimeOut

  Amount of time in seconds IBM Tivoli System Automation waits to restart resources which were located on a failed node on another node.
- TraceLevel

  The trace level can be used to control the amount of trace entries written. The maximum value of 255 results in very detailed tracing, while the value 0 suppresses writing various classes of trace entries. Reducing the trace level is advisable for automation policies with a large number of resources.

The current values of the attributes described above can be listed with the command **lssamctrl**. The attributes are changed with the **samctrl** command. Refer to *IBM Tivoli System Automation for Multiplatforms Base Component Reference* for a listing and description of these commands.

### TimeOut and RetryCount

The TimeOut and RetryCount parameter are always used together. They control how long IBM Tivoli System Automation will wait for a resource manager to do something and how many times it will tell the resource manager to try doing it again if it did not work the first time. In general, if it did not work the first time, the chances of it working on the second or subsequent attempts is fairly low.

### Start operations

The operation timer is started when IBM Tivoli System Automation first sends a resource start control operation to a resource. After that timer has started, there are 3 possibilities:

1. The resource changes to the desired state (online or offline) within the timeout period. With this the resource is in the state IBM Tivoli System Automation wants it to be, and no further actions are triggered.
2. The resource rejects the start control within the timeout period. What happens next depends on the reject code. If it indicates that the error is recoverable, IBM Tivoli System Automation will issue another start control operation against the resource. Every control operation try is counted and IBM Tivoli System Automation stops issuing control operations if the RetryCount is exceeded. If the error is not recoverable, the resource will go into a problem state. If it is a fixed resource nothing more will happen. If it is a floating resource, IBM Tivoli System Automation will try another instance, providing the instance that was tried to be started before is either Offline or Failed Offline. To get the resource out of the problem state, you should issue a reset operation against it.
3. The resource does not reach the desired state (online) within the timeout period. IBM Tivoli System Automation first issues a reset operation against the resource and waits until the reset operation has been accepted and the resource is offline. Then IBM Tivoli System Automation issues another start control operation against the resource. Every control operation try is counted and IBM Tivoli System Automation stops issuing control operations if the RetryCount is exceeded or the maximum timeout (which is TimeOut * RetryCount) expires, which ever comes first.

If IBM Tivoli System Automation stops issuing control operations for either a fixed resource or a constituent of a floating resource, the OpState of this resource is set to failed offline. This indicates that this resource is no longer usable. Now manual intervention is required, the cause of the failure must be removed, and the resource must be reset with the RMC resetrsrc command.

Note that no threshold is implemented, the retry counter is reset when the resource reaches its desired state. For example, if a resource is started, stays online for a short period of time, and then stops, it is restarted by IBM Tivoli System Automation in a loop.

Default values are:

- TimeOut = 60
- RetryCount = 3

You use the **samctrl –t** *Timeout* command to change the TimeOut value and **samctrl –r** *Retry_count* to change the RetryCount value.

The IBM.Application class provides its own timeout value. If you add such a resource to a group, the general TimeOut is not used for this resource. As TimeOut for this group member the larger value of StartCommandTimeout or MonitorCommandPeriod attribute (which are attributes of the IBM.Application resource) is used.

## Stop operations

The operation timer is started when IBM Tivoli System Automation first sends a resource stop control operation to a resource. After the timer has started, there are three possibilities:

1. The resource changes to the desired state (offline) within the timeout period. No further actions are triggered.
2. The resource rejects the stop control within the timeout period. What happens next depends on the reject code:
   - If it indicates that the error is recoverable, IBM Tivoli System Automation issues another stop control operation against the resource.
   - If the error is not recoverable, the resource goes into a problem state. Manual intervention is required to get the resource out of the problem state.
3. The resource does not reach the desired state (offline) within the timeout period. In this case, IBM Tivoli System Automation first issues a reset operation against the resource and waits until the resource reaches its desired state (offline).

## Automation

This flag indicates if IBM Tivoli System Automation automation functionality is enabled or not. If automation is disabled, IBM Tivoli System Automation will stop control operation. The state of resources will remain unchanged.

Default value is AUTO mode, which means that automation is turned on.

You use **samctrl –M F** to enable automation, **samctrl –M T** to disable automation.

## ExcludedNodes

This is a list of nodes where IBM Tivoli System Automation will stop all resources on and move them to another node if possible. For example, you have floating resource A which can run on four nodes node05, node06, node07 and node08. It is a member of resource group RG_A. After you made the group online it is started on node05. If you add node05 to the list of excluded nodes, IBM Tivoli System Automation will stop the resource on node05 and restart it on one of the other nodes.

Caution: If you exclude a node and one or more mandatory members of a group cannot be restarted on another node, this may cause the whole group to stop.

Default value is an empty list, that means all nodes in the peer domain can be used.

You use **samctrl –u a** to add one or more nodes to the list of excluded nodes, **samctrl –u d** to delete nodes from that list and **samctrl –u r** to replace nodes in the list.

## ResourceRestartTimeout

The ResourceRestartTimeout is the amount of time in seconds IBM Tivoli System Automation waits to restart resources on an online node which were located on a failed node. The reason behind this is to give the resources or the failed nodes a chance to do some cleanup before the resources are moved to another system.

Default value is 5 seconds.

You use **samctrl –o** to specify the resource restart timeout.

You can use **samctrl -l** to specify the trace level. The trace level (TraceLevel ) can be used to control the amount of trace entries written. The maximum value of 255

results in very detailed tracing, while the value 0 suppresses writing various classes of trace entries. Reducing the trace level is advisable for automation policies with a large number of resources. The default value is 127.

## Examples

To list the current IBM Tivoli System Automation control parameters you use the **lssamctrl** command.

```
lssamctrl
```

Displaying IBM Tivoli System Automation Control Information:

```
Displaying IBM Tivoli System Automation Control Information:

SAMControl:
        TimeOut                = 60
        RetryCount             = 3
        Automation             = Auto
        ExcludedNodes          = {}
        ResourceRestartTimeOut = 5
        ActiveVersion          = [1.2.0.0,Tue 04 May 2004 12:30:48 PM EDT]
        Enable Publisher       = Disabled
        TraceLevel             = 127
```

To add a node node05 to the list of excluded nodes, this command is entered:

```
samctrl -u a node05
```

To set the RetryCount parameter to 5, this command is entered:

```
samctrl –r 5
```

# Automation policy management

A core element of IBM Tivoli System Automation is the policy where the definitions are made how the system behaves. The policy consists of resource groups, relationships between resources, and/or groups and equivalencies. The main task of an administrator is to maintain this policy and to ensure that it is correct and recreatable.

To maintain one or more policies you use the **sampolicy** command. The command can be used in two different flavors:

1. To save a policy and restore it at a later point in time.
2. To completely replace a policy.

The **sampolicy** command handles resource groups, relationships, equivalencies, IBM.Application, IBM.ServiceIP and IBM.Test resources, control parameters (samctrl), and IBM.TieBreaker resources.

## Using the sampolicy command to manage policies

The following sections describe how to manage policies using the sampolicy command. For detailed information, see the description of the *sampolicy* command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

### Saving a policy

IBM Tivoli System Automation uses an XML file to define the automation policy. See the 'Policy Reference' chapter in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference* how to define an XML policy file.

Use the sampolicy command to save the policy to a file named mypolicy.xml:

```
sampolicy -s /usr/xml/mypolicy.xml
```

### Activating, restoring, and replacing a saved configuration

You can use the sampolicy command to activate, restore, or replace a policy which, for example, is specified in an XML file named myPolicy.xml:

```
sampolicy -a /usr/xml/myPolicy.xml
```

### Deactivating a running policy

To deactivate the currently active policy, issue:

```
sampolicy -d
```

# Using requests to start and stop resource groups and resources

You use requests to start or stop resources groups without changing their NominalState or to start and stop individual resource group members.

You can submit start and stop requests from the operations console or from the command line.

To submit such a request against a resource group from the command line, you use the command **rgreq –o <start|stop>**. To start or stop a resource group member, you use the command **rgmbrreq –o <start|stop>**.

Start and stop requests are persistent, which means that you must cancel them to remove them from a resource's request list. When an active request is canceled and there are other requests in the request list, the request ranking next on the list is activated. To cancel requests, you use the command **rgreq –o cancel** or **rgmbrreq –o cancel**.

Requests always overrule the NominalState value of a resource group.

To prioritize requests, the base component utilizes the priority level and the value of the source attribute. Depending on how a request is submitted, the Source attribute and the priority level of the request are set to default values. You can override the default values when you issue the commands **rgreq** and **rgmbrreq** by using the **–S** option to specify the source and the **-p** option to set the priority level of the request. For more information on how requests are prioritized, see “Sources and default priorities of start and stop requests” on page 173 and “How requests are prioritized” on page 174.

Start and stop request from the same source replace each other because the request list for a resource group or managed resource can only contain one request from each source.

You can use the **lsrgreq** command to display submitted requests. The example scenario below illustrates how an operator changes the operational state of a resource group by issuing and canceling requests and how the **lsrgreq** command can be used to display a resource's request list and to track the success of requests.

**Example scenario**

The NominalState of the group "top-rg" is Offline. The OpState of the group, however, is Online. The output of the **lsrgreq** command below shows that the group's OpState results from an Active start request from end-to-end automation:

```
lnxcm3x:# lsrgreq -L -g top-rg
Displaying Resource Group request information:
All request information
For Resource Group "top-rg".

Resource Group 1:
        ResourceGroup = top-rg
        Priority      = High
        Action        = start
        Source        = Automation
        NodeList      = {}
        ActiveStatus  = Active
        Token         = 8f5697eb5f84c0f044995b3d00040a5b
        UserID        =
        MoveStatus    = None
```

```
lnxcm3x:# lsrg -g top-rg | grep State
Displaying Resource Group information:
For Resource Group "top-rg".

Resource Group 1:
        NominalState         = Offline
        OpState              = Online
        TopGroupNominalState = Offline
```

In an attempt to bring the group "top-rg" offline, an operator issues a stop request:

```
rgreq –o stop top-rg
```

Now the request list for "top-rg" looks like this:

```
lnxcm3x:# lsrgreq -L -g top-rg
Displaying Resource Group request information:
All request information
For Resource Group "top-rg".


Resource Group 1:
        ResourceGroup = top-rg
        Priority      = High
        Action        = start
        Source        = Automation
        NodeList      = {}
        ActiveStatus  = Active
        Token         = 8f5697eb5f84c0f044995b3d00040a5b
        UserID        =
        MoveStatus    = None

Resource Group 2:
        ResourceGroup = top-rg
        Priority      = low
        Action        = stop
        Source        = Operator
        NodeList      = {}
        ActiveStatus  = InActive
        Token         = 8f5697eb5f84c0f044995dad0007b338
        UserID        =
        MoveStatus    = None
```

The output of the **lsrgreq** command above shows that the operator's stop request was added to the request list but did not win. The stop request remains inactive because it was issued with the default priority (low) and could not win against the higher ranking start request from end-to-end automation:

```
lnxcm3x:# lsrg -g top-rg | grep State
Displaying Resource Group information:
For Resource Group "top-rg".

Resource Group 1:
        NominalState         = Offline
        OpState              = Online
        TopGroupNominalState = Offline
```

Now the operator issues a high-priority stop request to bring the "top-rg" group offline:

```
rgreq –p high –o stop top-rg
```

The output of the **lsrgreq** command below shows that this request did override the operator's previous low-priority request, won against the Online request from end-to-end automation, and has become the active request:

```
lnxcm3x:# lsrgreq -L -g top-rg
Displaying Resource Group request information:
All request information
For Resource Group "top-rg".

Resource Group 1:
        ResourceGroup = top-rg
        Priority      = High
        Action        = start
        Source        = Automation
        NodeList      = {}
        ActiveStatus  = InActive
        Token         = 8f5697eb5f84c0f044995b3d00040a5b
        UserID        =
        MoveStatus    = None

Resource Group 2:
        ResourceGroup = top-rg
        Priority      = High
        Action        = stop
        Source        = Operator
        NodeList      = {}
        ActiveStatus  = Active
        Token         = 8f5697eb5f84c0f044996004000368b1
        UserID        =
        MoveStatus    = None
```

The group "top-rg" is stopped due to the stop request:

```
lnxcm3x:# lsrg -g top-rg | grep State
Displaying Resource Group information:
For Resource Group "top-rg".

Resource Group 1:
        NominalState         = Offline
        OpState              = Offline
        TopGroupNominalState = Offline
```

To bring the "top-rg" group Online again, the operator must only cancel the last Offline request, an additional Online request is not required:

```
rgreq –o cancel top-rg
```

The output examples below show that the "top-rg" group is started after the request is canceled because the request from end-to-end automation becomes active again:

```
lnxcm3x: # lsrgreq -L -g top-rg
Displaying Resource Group request information:
All request information
For Resource Group "top-rg".


Resource Group 1:
        ResourceGroup = top-rg
        Priority      = High
        Action        = start
        Source        = Automation
        NodeList      = {}
        ActiveStatus  = Active
        Token         = 8f5697eb5f84c0f044995b3d00040a5b
        UserID        =
        MoveStatus    = None

lnxcm3x:# lsrg -g top-rg | grep State
Displaying Resource Group information:
For Resource Group "top-rg".
```

```
Resource Group 1:
Resource Group 1:
        NominalState         = Offline
        OpState              = Online
        TopGroupNominalState = Offline
```

For a detailed description of the commands mentioned above, refer to the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

## Scope of start and stop requests

The scope of a start or stop request depends on whether it is submitted against a resource group or against a specific member of a resource group:

- A request that is submitted against a resource group affects all members of the group. Additional resources may be affected if relationships are defined in the policy for the resource group or any of its members or both.
- A request that is submitted against a specific member of a resource group only affects the particular resource. Additional resources may be affected when the resource has relationships.

**Note:** A request that is issued against a mandatory member of a resource group may also have an impact on the resource group's observed and composite states. This happens when the request brings the member into an observed state that conflicts with the desired state of the resource group. In this case, the observed state of the resource group changes to a "pending" state and its composite state changes to "error".

**Example:**

When an operator submits a stop request against a mandatory member of a resource group whose desired state is Online, the observed state of the resource group changes to "pending online" and its composite state changes to "error" when the member resource is stopped.

## Sources and default priorities of start and stop requests

The default value to which the Source attribute of a request is set and the request's default priority level depend on how the request is submitted (Submitter) and on the type of resource against which it is issued (Target resource). Table 20 shows the default values.

*Table 20. Sources and priority levels of start and stop requests*

| Submitter | Target resource | Source attribute value | Default priority level |
|---|---|---|---|
| Operations console in any mode | Resource or resource group | Operator | high |
| Base component command-line interface | Resource or resource group | Operator | low<br><br>Possible values: low, high, force |
| External scheduler (for example, Tivoli Workload Scheduler or a cron job) | Resource or resource group | ExtSched | High<br><br>Possible values: low, high, force |

*Table 20. Sources and priority levels of start and stop requests (continued)*

| Submitter | Target resource | Source attribute value | Default priority level |
|---|---|---|---|
| End-to-end automation engine | End-to-end automation resource reference that references a base component resource or resource group | Automation | high |
| Operations console in end-to-end automation mode | End-to-end automation resource reference that references a base component resource or resource group | Automation | high |
| End-to-end automation manager command line interface | End-to-end automation resource reference that references a base component resource or resource group | Automation | high |

## How requests are prioritized

Requests that are submitted against a resource group are prioritized in the following way:

- Any request takes precedence over the NominalState of the resource group.
- If requests have different priority levels, the priority level determines a request's priority rank.
- If requests have the same priority level, the Source attributes are evaluated to determine the priority rank of each request. Figure 18 illustrates how the Source attribute value influences the priority ranking of requests.

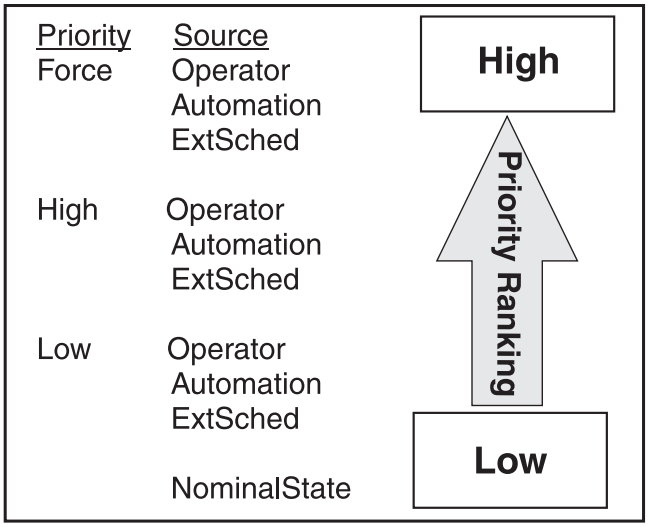



*Figure 18. Priority ranking of requests*

Figure 18 shows that requests from source 'Operator' take precedence over requests from sources 'Automation' and 'ExtSched' and requests from Source

'Automation' take precedence over requests from source 'ExtSched'. This means that you can overrule requests from source 'Automation' by issuing command line requests using the option **-p high**. To overrule a request from source 'Operator', you must specify the option **-p force**.

## Priority of operations console requests

All online and offline request that are submitted from the base or end-to-end automation operations console result in start or stop requests from source 'Automation'. Such requests have a higher priority than the NominalState. Operators can overrule such requests from the command line by submitting commands with the priority 'Force'.

# Moving resource groups with the rgreq command

There are situations where the user would like the capability to move individual resource groups to another cluster node without affecting all resources currently running on the same node. For example, in a load-balancing situation, moving only one or several resource groups to another node may achieve the user's workload and performance objectives. Adjustments to the placement of resources can be done using the **rgreq –o move** command.

## Scope of a move

The scope of a move is all the members of a top level resource group. Resources that are dependent upon one or more resources involved in the move might be affected, i.e. stopped and started. A move request cannot be issued against a single managed resource.

If the MemberLocation attribute of the top level resource group is set to collocated, no list of nodes has to be provided with the rgreq command. In this case, all resources are located on the same node and will be moved away from that node. If the resource group is not collocated, a list of one or more nodes has to be specified with the –n option of the rgreq command. All resources will be moved away from those nodes.

A move request issued against a resource group that contains only fixed resources will not be accepted. Also, a move request issued against an offline resource group will be refused. Once a move request is accepted, it cannot be cancelled by the issuer. While there is a move request already in progress, another move request on the same resource group will be rejected.

## Processing of a move request

An offline phase is processed in the course of the move request, in which all the members of the moved resource group will be stopped first, even if they are running on nodes that are not contained in the list of nodes to move away from. This is done to avoid complications when placing the resources later on. The binder assigns a new node placement for all resource group members, and the resource group is restarted. This is the online phase of the move. Note that if it turns out to be impossible to restart the mandatory members of the top level resource group while honouring the list of nodes to move away from, the list will be ignored and resources may be restarted upon it. Likewise, if the only place to restart a resource is the original system it was running on, then it will be restarted there if it is a mandatory resource.

A move request is automatically removed when the move action is carried out. The MoveStatus dynamic attribute of the resource group being moved will show values indicating the progress of the move.

## Move and relationships

In addition to performing a move request on the members of a top level resource group, there may be other resource groups and/or resource group members outside the moved top level group, which placement must be adjusted according to defined relationships constraints. This applies to the following relationships:

- Collocation
- AntiCollocation
- DependsOn

- DependsOnAny
- StopAfter
- ForcedDownBy

In addition, the Affinity and AntiAffinity relationships might not be fulfilled after the move.

## Using shadow resources

A shadow resource has the following characteristics:

- It is a fixed resource of class IBM.Application that monitors the OpState of another fixed resource.
- It is member of an equivalency whose SelectFromPolicy attribute value is set to NoControl, which specifies that IBM Tivoli System Automation only evaluates the OpState of the member resources but does not start or stop them.

You must use shadow resources if you want to define a relationship of type DependsOn from a floating resource to one out of two or more fixed resources that are running on different nodes, because the DependsOn relationship triggers both a StartAfter and a StopAfter behavior.

If you set up such a DependsOn relationship without defining shadow resources, an undesired automation behavior occurs:

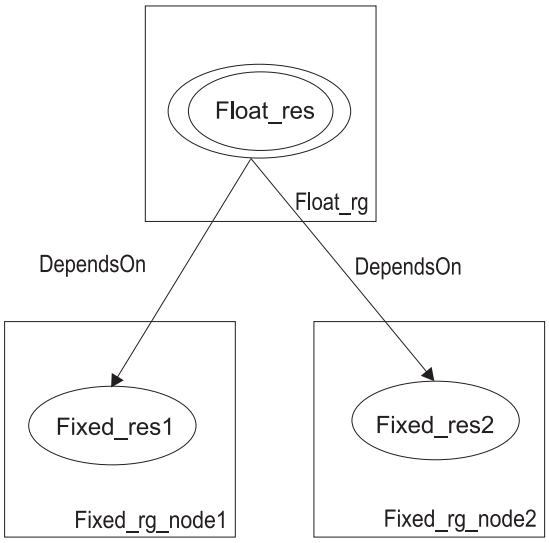



*Figure 19. Scenario 1: DependsOn relationship, without shadow resources*

In Scenario 1, IBM Tivoli System Automation will stop the floating resource even if only one of the fixed resources is not available, and IBM Tivoli System Automation can only restart the floating resource if both of the fixed resources are available.

To achieve the desired automation behavior, you create shadow resources that are members of a NoControl equivalency:

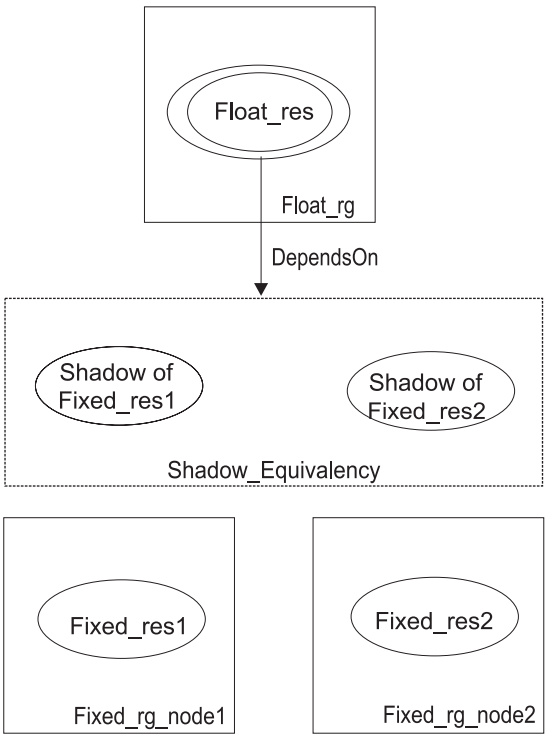



*Figure 20. DependsOn relationship, with shadow resources*

In Scenario 2, the ForceDown behavior of the DependsOn relationship will still cause IBM Tivoli System Automation to stop the floating resource if the operational state of the currently selected member of the equivalency changes to Offline, but now IBM Tivoli System Automation can restart the floating resource again if at least one other member of the equivalency is online.

## Defining shadow resources

To create shadow resources, perform the following steps:

1. Create the command scripts for the shadow resources.

   Although IBM Tivoli System Automation will only use the MonitorCommand, both a valid StartCommand and a valid StopCommand must also be supplied. The MonitorCommand must query the OpState of the resource it is a shadow for. This can be achieved in one of the following ways:

   - by querying the OpState using

     ```
     'lsrsrc –s 'Name like "<res> "'IBM.Application OpState
     ```

   - by duplicating the MonitorCommand of the original resource
   - by saving the OpState of the original resource to a file and reading it within the MonitorCommand of the shadow resource

   **Example:**

   The following sample script contains the required commands for the shadow resource of the resource “fixed_rs1”:

   ```
   #!/bin/ksh
   #
   # shadow_sample.sh
   #
   # init section
   #

   Action=${1:-status}
   ```

```
ResName=${2:-myresource}

UNKNOWN=0
ONLINE=1
OFFLINE=2
FAILED_OFFLINE=3

export CT_MANAGEMENT_SCOPE=2 # necessary to execute SA MP commands

#
# main section
#
case ${Action} in
  start)
         # is not executed .. so irrelevant
         RC=0
         ;;
  stop)
         # is not executed .. so irrelevant
         RC=0
         ;;
status|*)
RCval=$(lsrsrc -xt -s 'Name="'${ResName}'"
   IBM.Application OpState)
         RCx=${RCval:-2}
         case ${RCx} in
           [1]*)   RC=${ONLINE}
                   ;;
           *)      RC=${OFFLINE}
                   ;;
         esac
         #logger -i -t "$(basename $0)" "${ResName} monitored: ${RC}"
esac

exit ${RC}
```

2. Create the shadow resources, using a command like the following for each of the resources:

```
# mkrsrc IBM.Application \
 Name="fixed_rs1_shadow" \
 ResourceType=0 \
 NodeNameList="{'node1'}" \
 UserName="root" \
 StartCommand="/samplepath/shadow_sample.sh start fixed_rs1" \
 StopCommand="/samplepath/shadow_sample.sh stop fixed_rs1" \
 MonitorCommand="/samplepath/shadow_sample.sh status fixed_rs1" \
 MonitorCommandPeriod=10 \
 RunCommandsSync=1
```

3. Create an equivalency with a SelectFromPolicy attribute value of NoControl.

   To create the equivalency containing the fixed resources created in step 2, use the following command:

```
# mkequ <equ-name> -p A,NoControl \
   IBM.Application:<fixed-resource1>:<node-name1>,<fixed-resource2>:<node-name2>[,...]
```

   **Example:**

```
mkequ -p A,NoControl shadow_equ IBM.Application:fixed_rs1_shadow:node1,fixed_rs2_shadow:node2
```

   To define an ordered policy, use `-p O,NoControl`.

4. Create the DependsOn relationship from the floating resource to the equivalency 'shadow_equ':

```
# mkrel –p DependsOn –S IBM.Application:float1 –G IBM.Equivalency:shadow_equ \
            float1-depon-shadow_equ
```

## Setting up non-root security

This section describes how you set up non-root security for the command line interface of the base component of IBM Tivoli System Automation.

By default, only the user `root` has the required authority to perform operational tasks in IBM Tivoli System Automation, or to make changes to the policy of IBM Tivoli System Automation, while all other users only have read access.

The security concept of IBM Tivoli System Automation is based on that of the RSCT component RMC, which implements security authorization using an access control list (ACL) file. Specifically, RMC uses the ACL file on a particular node to determine the permissions that a user must have in order to access resource classes and their resource instances. Since the System Automation resource managers are internally implemented as an RMC application, the same set of ACL control rules must be used in order to allow non-root users to manage (define, undefine, or change) the System Automation related resource classes (IBM.ResourceGroup, IBM.ManagedRelationship, IBM.Equivalency, IBM.ManagedResource, IBM.CHARMControl, IBM.Application and IBM.ServiceIP) and to start and stop the corresponding resource groups.

For detailed information on how to set up RMC ACL files, see the following sections in the *IBM RSCT Administration Guide*:

- “Managing user access to resources using RMC ACL files” in Chapter 4 ("Managing and monitoring resources using RMC and resource managers")
- “Configuring the global and local authorization identity mappings” in Chapter 7 ("Understanding and administering cluster security services")

RSCT and RMC security authorization support is capable of managing user access based on individual resource classes and single nodes, for example, user access can be restricted to a specific RMC resource class on a particular node within the cluster. This level of authorization setting, however, is very complex and requires a clear understanding of the nature of each individual RMC resource class. Therefore it is recommended that you create roles for an IBM Tivoli System Automation operator and an IBM Tivoli System Automation administrator with general settings that will allow non-root users to manage all resource classes from any node defined within the cluster. By using the procedure described below you will create these two roles:

- “sa_admin” for an administrator
- “sa_operator” for an operator

To create the roles, perform the following steps (note that root authority is required):

1. Create the user IDs that are to be authorized to manage IBM Tivoli System Automation on all nodes. Use the appropriate command for your operating system, for example, on Linux use:

   ```
   # /usr/sbin/useradd ernie
   # /usr/sbin/useradd bert
   ```

2. Create a group for the user IDs on all nodes. Use the appropriate command for your operating system, for example, on Linux use:

   ```
   # /usr/sbin/groupadd sagroup
   ```

3. Add the group to the user IDs on all nodes. Use the appropriate command for your operating system, for example, on Linux use:

```
# /usr/sbin/usermod –G sagroup ernie
# /usr/sbin/usermod –G sagroup bert
```

**Note:** Make sure to set the following environment variable for all users of IBM Tivoli System Automation on all nodes (peer domain scope):
`CT_MANAGEMENT_SCOPE=2`
You can set the variable permanently if you set it in the profile.

4. Change the group ownership of the file `/var/ct/IBM.RecoveryRM.log`.

   The file is used to track the IBM Tivoli System Automation history. All commands that modify the resources of the automation manager (IBM.RecoveryRM) are logged to that file.

   By default, the file is owned by the user group `root`:

   ```
   -rw-r--r--  1 root root   204 Oct 4 22:00 /var/ct/IBM.RecoveryRM.log
   ```

   You need to change the group ownership to “sagroup”, using the appropriate command for your operating system, for example, on Linux use:

   ```
   /bin/chgrp sagroup /var/ct/IBM.RecoveryRM.log
   ```

   You also need to change the file permission to `664`:

   ```
   # /bin/chmod 664 /var/ct/IBM.RecoveryRM.log
         -rw-rw-r-- 1 root sagroup 204 Oct 4 22:00 /var/ct/IBM.RecoveryRM.log
   ```

   **Note:** If the file `/var/ct/IBM.RecoveryRM.log` does not exist after the initial installation of IBM Tivoli System Automation, you can create a dummy file using the ***/usr/bin/touch*** command:

   ```
   # /usr/bin/touch /var/ct/IBM.RecoveryRM.log
   ```

5. Create or modify the file `/var/ct/cfg/ctsec_map.global` on all nodes.

   You need to add the following entries for the user IDs “ernie” and “bert” to the RSCT global authorization identity mapping file (`/var/ct/cfg/ctsec_map.global`) on every node in the cluster:

   ```
   unix:ernie@<cluster>=sa_operator
   unix:ernie@<any_cluster>=sa_operator
   unix:bert@<cluster>=sa_admin
   unix:bert@any_cluster>=sa_admin
   ```

   The file is used to map a local user ID on a node to a global user ID within the IBM Tivoli System Automation domain. In the example, the local user ID “ernie” is mapped to the global user ID “sa_operator”, and the local user ID “bert” is mapped to the global user ID“sa_admin”.

   You can easily authorize additional local user IDs for IBM Tivoli System Automation by adding additional lines to this global map file (on all nodes), and mapping these to the desired role (IBM Tivoli System Automation Operator or IBM Tivoli System Automation Administrator).

   **Note:** If the file `/var/ct/cfg/ctsec_map.global` does not exist on a node, copy the default file `/usr/sbin/rsct/cfg/ctsec_map.global` to the directory `/var/ct/cfg`, and add the new entries to the file `/var/ct/cfg/ctsec_map.global`. Do not remove any entries from the file `/var/ct/cfg/ctsec_map.global` that exist in the default file you copied. The `/var/ct/cfg/ctsec_map.global` files on all nodes within the cluster must be identical.

6. Modify the file `/var/ct/cfg/ctrmc.acls` on all nodes. You need to add the following entries for the global user IDs “sa_operator” and “sa_admin” to the RMC ACL file (`/var/ct/cfg/ctrmc.acls`) on every node in the cluster and comment the line starting with `LOCALHOST`, for example:

```
# The following stanza contains default ACL entries.
# These entries are appended
# to each ACL defined for a resource class and
# are examined after any entries
# explicitly defined for a resource class
# by the stanzas in this file,
# including the OTHER stanza.

DEFAULT
 root@LOCALHOST       *    rw
 # LOCALHOST       *     r  // comment this line out!
 none:root   *  rw   // give root access to all
 none:sa_admin   *  rw   // append this row for saadmin
 none:sa_operator   *  rs  // append this row for saoperator
```

7. Once the above modifications are complete the following command must be executed on every node in the cluster to activate the changes:

   ```
   # /usr/bin/refresh -s ctrmc
   ```

8. Additional changes required to use sampolicy and *samadapter commands:

   a. Access to configuration files:

      ```
      # /bin/chgrp -R sagroup /opt/IBM/tsamp/sam/cfg
      # /bin/chmod g+ws /opt/IBM/tsamp/sam/cfg
      # /bin/chmod g+w /opt/IBM/tsamp/sam/cfg/*
      ```

   b. Access to log files:

      ```
      # /bin/chgrp -R sagroup /var/ibm/tivoli/common/eez/logs
      # /bin/chmod g+ws /var/ibm/tivoli/common/eez/logs
      # /bin/chmod g+w /var/ibm/tivoli/common/eez/logs/*
      ```

   c. Access to configuration files in the `/etc` directory.
      If there is no directory `/etc/opt/IBM/tsamp/sam/cfg`, create it by using

      ```
      # /bin/mkdir -p /etc/opt/IBM/tsamp/sam/cfg
      ```

      Then set the permissions appropriately:

      ```
      # /bin/chgrp -R sagroup /etc/opt/IBM/tsamp/sam/cfg
      # /bin/chmod g+ws /etc/opt/IBM/tsamp/sam/cfg
      # /bin/chmod g+w /etc/opt/IBM/tsamp/sam/cfg/*
      ```

9. Optional adjustments required for working with the “sam.policies ” package:
   Pre-canned policies for various applications are provided in the install package “sam.policies ”, which can be downloaded from the following FTP location:

   ```
   ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux
   ```

   To allow a user who has the “sa_admin” role to set up these pre-canned policies, the permissions and the ownership of the `/usr/sbin/rsct/sapolicies` directory must be changed after the “sam.policies ” package is installed on all nodes:

   ```
   # chmod –R 2775 /usr/sbin/rsct/sapolicies
   # chgrp -R sagroup /usr/sbin/rsct/sapolicies
   ```

**Results:** When the above steps have been completed successfully, the local users “ernie” and “bert” can perform operational tasks of IBM Tivoli System Automation, such as issue start and stop requests against resources, and the local user “bert” can also perform administrative tasks of IBM Tivoli System Automation, such as define and modify policies.

## Limitations of the non-root security setup

The following list summarizes the limitations of the non-root security setup described above:

- Regular user cannot view the contents of the RMC resource manager trace file (for example, the trace of the IBM.RecoveryRMd daemon).

All RMC Resource Manager daemons use the RMC framework library utility to create trace files and core images under the /var/ct/<cluster> directory. Since these resource managers can only be started by a superuser (user ID root) through the **/usr/bin/startsrc** command, the files that are created belong to the user ID root.

This implies that a non-root user cannot collect debug and trace information by using the **/usr/sbin/rsct/bin/ctsnap** command.

To allow non-root users to collect traces or ctsnap debug data or both, a mechanism like "sudo" must be implemented for these users and commands.

- The following commands can only be invoked with root authority because they utilize Tivoli logging, which only works properly if the log files are maintained with root rights:
  - The **sampolicy** command
  - The **samadapter** command to start the end-to-end automation adapter
- The granularity of the ACL objects is based on resource classes, not on resources. This means that a regular user is either allowed to modify resources of a resource class or not, but it is not possible to grant or deny permissions on a resource basis, for example, a database administrator cannot be authorized only for database resources.
- The "sa_operator" role can modify resources by changing attribute values for the resources. This is a result of the "s" permission, which is needed for issuing IBM Tivoli System Automation requests. Without the "s" permission, users who have this role would not be able to perform any useful task, but with this permission, they are allowed to change and set attributes.

The following table shows which role or authority is required to perform typical IBM Tivoli System Automation tasks.

*Table 21. Authorizations and roles for performing IBM Tivoli System Automation tasks*

| Task | Authority | Roles | Permissions |
|---|---|---|---|
| Product installation | root | System Administrator | Installing and upgrading IBM Tivoli System Automation |
| Cluster management | root / sa_admin | System Administrator / IBM Tivoli System Automation Administrator | Defining, starting, stopping, and monitoring clusters and individual RMC Resource Managers |
| Resource definition and IBM Tivoli System Automation policy definition | root / sa_admin | System Administrator / IBM Tivoli System Automation Administrator | Defining, removing, changing resources, and setting up automation policies |
| Automation operation | root / sa_admin / sa_operator | System Administrator / IBM Tivoli System Automation Administrator and Operator | Issuing Online and Offline request, and resetting and monitoring resource groups and individual resources |
| Collecting trace and debug data for problem determination | root | System Administrator | Access to all system and application trace (log) files. (See the list of limitations above.) |

*Table 21. Authorizations and roles for performing IBM Tivoli System Automation tasks (continued)*

| Task | Authority | Roles | Permissions |
|---|---|---|---|
| Security setup | root | System Administrator | Defining, changing, and removing the security setup described in this section. |
| Adapter setup | root / sa_admin | System Administrator / IBM Tivoli System Automation Administrator | Defining, changing, and removing the configuration of the end-to-end automation |

# Diagnosing IBM Tivoli System Automation Resources

To get more information about resources managed by IBM Tivoli System Automation you can use the **samdiag** command, which is documented the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*. This command is mainly intended to be used in situations where it is not obvious to the user what is happening on the system and why.

**Note:** The samdiag command provided for Release 1 of IBM Tivoli System Automation could only be executed on the node where the master daemon was running on. So running samdiag on a Release 2 daemon will generate an error if Release 1 and Release 2 daemons coexist in the same cluster, and if the master daemon is on a Release 1 node.

To get information about a resource group called ″apacherg″, you use the command:

```
samdiag -g apacherg
```

Output:

```
Diagnosis::Resource: apacherg/ResGroup/IBM.ResourceGroup
    type: CHARM Resource Group
  Status -
      Observed: Offline         - SoftDown
        Desired: Offline         - Requested Offline
       (Nominal: Offline         - Nominal State: Offline)
     Automation: Idle            - CharmBase trigger linked
      Startable: Yes             - Resource is startable
        Binding: Unbound         - Binding status initialized
       Compound: Satisfactory    - Satisfactory

 Resource Based Quorum: None            -
   Members and Memberships:
       +---bind/HasMember                ---> RA/Float/IBM.Test
   Group Constraint: None
   Binding Constraints:
   Flags:
     None
   Orders:
     Outstanding Order: None            - Resource is Unavailable
   Dependencies:
    Start: Satisfied
       +---InCluster                     ---> Cluster
     Stop: Satisfied
   Binding exceptions:
     There are unbound members.
   Static Relationships:
       +---InCluster                     ---> Cluster
   Dynamic Relationships:
       +---bind/HasMember                ---> RA/Float/IBM.Test"
```

The following provides an interpretation of some of the information given in the example:

- The ObservedState should show the same value as OpState and NominalState. If this is not the case, contact the support center serving your location.
- Different values for DesiredState and NominalState indicate that a request has been issued against the resource group.
- The AutomationState can be 'busy' or 'idle'. 'Busy' means that the IBM Tivoli System Automation daemon (IBM.RecoveryRM) is waiting for another resource

manager to start or stop a resource. After this has been completed the AutomationState changes to 'idle'. If this is not the case, contact the support center serving your location.

- A Startable state of 'No' indicates that some relationships, for example a DependsOn relationship, are not correctly set.
- A BindingState of 'Unbound' comes along with a Observedstate 'Offline'. It indicates that the resource group is offline. Before it can be set to online a binding step must be performed which chooses the correct constituent. Then the BindingState is set to 'Bound'. A BindingState of 'Bound' for resource groups being offline is an error.
- A CompoundState of 'Satisfactory' indicates that ObservedState and DesiredState are the same. CompoundStates 'Inhibited', 'Denied', or 'Broken' indicate errors like relationships which have not been fulfilled or 'broken' resources.
- Resource Based Quorum equals 'None' means that all floating resources of the group do not support a Resource Based Quorum of their own and the Resource Based Quorum flag is not set.
- 'Bind/HasMember' describes the relationship between resource group apacherg and its floating member RA/Float/IBM.Test. When performing the binding step mentioned above a constituent is selected for each of the 'Bind/HasMember' relationships. This constituent is temporarily bound to the resource group before this resource group is set to online.
- Outstanding Orders refer to the AutomationState. If AutomationState is not 'idle', the pending command is shown here.
- Start/Stop Dependencies show when a policy prevents the start of a resource.
- Binding exceptions provide a closer explanation of BindingState.
- Static relationships mean that all constituent resources and resource groups are members of the cluster.
- Dynamic relationships are temporary relationships caused by the binding step.

# Using the IBM Tivoli System Automation TEC event interface

Whenever IBM Tivoli System Automation changes its configuration, resource status or encounters any problems, the Tivoli Enterprise Console (TEC) event interface can be used to notify the system administrator. There are two ways to notify the administrator:

1. Sending events to the IBM Tivoli Enterprise Console. As a prerequisite the TEC publisher function must be enabled (see "Enabling the TEC publisher function").
2. Publishing IBM Tivoli System Automation internal attributes into the RSCT infrastructure. The administrator must subscribe to one of the Event Resource Manager (ERRM) scripts in order to get event information. Refer to the manual *IBM Reliable Scalable Cluster Technology for Linux,Technical Reference*, SA22–7983, how to do this; also see the description in the preceding section "Generating events in case of failures" on page 196.

## What is the Tivoli Enterprise Console?

The Tivoli Enterprise Console (TEC) is a rule-based event management application that uses a central server to process incoming events. TEC acts as a central collection point for alarms and events from a variety of sources, including those of Tivoli applications, Tivoli partner applications, customer applications, network management platforms, and relational database systems.

## What are events?

Events are units of information which can represent performance data or also can indicate problems, status or changes regarding resources. Usually IBM Tivoli System Automation sends events when the help of an administrator is required.

A language called Basic Recorder of Objects in C (BAROC) is used to define the structure of events and their properties. These definitions are stored in files with the extension **.baroc**.

## Sending events to the TEC

IBM Tivoli System Automation uses the TEC event interface 'Tivoli Event Integration Facility (EIF)' to send events to the TEC. The events will be sent to the TEC server non-TME event port.

The following reasons cause IBM Tivoli System Automation to send events to the TEC:

- The status of a resource or cluster changed.
- A resource was added or deleted.
- A relationship was added or deleted.
- A request was added or deleted.

## Enabling the TEC publisher function

In order to receive and view the events coming from different sources like programs, systems or network devices you have to enable the TEC publisher by performing the following steps:

1. Import, compile, load, and activate the TEC baroc file in the TEC server (/usr/sbin/rsct/samples/tec/SystemAutomation.baroc). Refer to manual *IBM Tivoli Enterprise Console Rule Builder's Guide* , GC32–0669, how to do this.

2. Copy files */usr/sbin/rsct/samples/tec/samPublisher.conf* and */usr/sbin/rsct/samples/tec/TECPublisher.conf* into */etc/Tivoli/tec* on any IBM Tivoli System Automation cluster node.
3. Customize the publisher configuration file */etc/Tivoli/tec/samPublisher.conf* and the TEC EIF file */etc/Tivoli/tec/TECPublisher.conf* on each cluster node.
4. Enable the publisher with the command **samctrl –e P** on a IBM Tivoli System Automation cluster node. By default the publisher is disabled.

## Publisher configuration file /etc/Tivoli/tec/samPublisher.conf

The publisher configuration file specifies a list of all target consumers and their parameters. This is the syntax format of the publisher configuration file:

```
#
# Publisher configuration file
# file name: /etc/Tivoli/tec/samPublisher.conf
#
# File format:
#    <keyword>=<value>
#
#   Publisher     - unique name of the publisher
#                   name length: 1-8 characters
#                   valid characters: '0'-'9', 'A'-'Z', 'a'-'z' and '_'
#   LibraryPath   - name of the publisher library
#   ConfigPath    - full path to the TEC EIF configuration file
#
# Multiple entries of the Publisher, LibraryPath and ConfigPath can be specified.
# One triplet for each publisher target consumer.
# Maximum supported publishers: 15

# Online Update section ------------------------------------------------------
# End Online Update section --------------------------------------------------

Publisher=TEC
LibraryPath=libTECPublisher.so
ConfigPath=/etc/Tivoli/tec/TECPublisher.conf

# Publisher=TEC2
# LibraryPath=libTECPublisher.so
# ConfigPath=/etc/Tivoli/tec/TECPublisher2.conf
```

*Figure 21. Syntax format and sample of the publisher configuration file*

The following syntax rules apply:

- lines starting with # and blank lines will be ignored
- parameter format: <keyword>=<value>
- keyword ″Publisher″ starts a new triplet of ″Publisher″, ″LibraryPath″ and ″ConfigPath″ parameters
- keyword ″Publisher″ specifies the unique name of the publisher
- keyword ″LibraryPath″ specifies the full path to the publisher library
- keyword ″ConfigPath″ specifies the full path to the TEC EIF configuration file
- keyword "Publisher" value must have the length of 1-8 characters
- keyword "Publisher" value can only have the following characters: '0'-'9', 'A'-'Z', 'a'-'z' and '_'
- the maximum of 15 publishers can be specified in this configuration file

## TEC EIF configuration file /etc/Tivoli/tec/TECPublisher.conf

The TEC EIF configuration file specifies all parameters needed to connect to a specific TEC server. The file name must match the name specified as ″ConfigPath″ parameter in the publisher configuration file.

The syntax format of the TEC EIF file for the TEC publisher is the existing TEC EIF configuration file syntax.

```
# TEC EIF configuration file
#
# File format:
#   <keyword>=<value>
#
#   ServerLocation  - name of the host where the TEC server is running
#   ServerPort      - port number on which the TEC server is listening
#                     to non-TME TEC events. TME TEC events are not supported
#                     5529 - default non-TME port for TEC servers on Windows
#                     0    - default non-TME port for TEC servers on AIX and Linux
#   ConnectionMode  - distinguishes between connection_oriented OR connection_less
#                   - (default is connection_oriented)
#   BufferEvents    - specifies whether the event buffering cache file is enabled
#                             (YES | MEMORY_ONLY | NO) (default is YES)
#   BufEvtPath      - specifies the full path name of the cache file
#                          (default: /etc/Tivoli/tec/cache)
#   NO_UTF8_CONVERSION  - Specifies if UTF8 conversion is done again in EIF library
#                         Must be YES, otherwise event is corrupted
#                        (default is NO)
#   FilterMode      - specifies whether events that match a Filter are  sent to
#                     the event server (FilterMode=IN) or are discarded
#                     (FilterMode=OUT)  (default is OUT)
#   Filter          - Filter:Class=class_name;[attribute=value[;attribute=value]*]
#
# For a description of all supported keywords and their values refer to manual:
# "Tivoli Event Integration Facility - Reference", SC32-1241,
# Chapter: "Appendix B. Keywords for Configuration Files".

#Put the server name or IP address of the server on which the TEC is running into
#the "ServerLocation" field.
ServerLocation=tecserver.ibm.com
ServerPort=5529
ConnectionMode=connection_less
BufferEvents=YES
BufEvtPath=/etc/Tivoli/tec/TECPublisher.cache
NO_UTF8_CONVERSION=YES

# Default Filters
# Filter all relationship add / delete events
Filter:Class=SystemAutomation_Relationship_Configuration_Change
# Filter all resource add / delete events
Filter:Class=SystemAutomation_Resource_Configuration_Change
# Filter resource status events with severity HARMLESS
Filter:Class=SystemAutomation_Resource_Status_Change;severity=HARMLESS
# Filter resource status events with severity WARNING
Filter:Class=SystemAutomation_Resource_Status_Change;severity=WARNING
# Filter all request add / delete events
Filter:Class=SystemAutomation_Request_Configuration_Change
```

*Figure 22. Syntax format and sample of the TEC configuration file*

To avoid that the TEC is flooded with a huge amount of messages, filters are provided in the # TEC EIF configuration file. Per default all filters are enabled,

which results in only critical messages being sent to the TEC. If you want additional messages to be sent to the TEC, disable the corresponding filter by using the comment character #.

### Enabling the publisher

Per default the Publisher function is disabled. To query the status of the publisher issue the following command:

```
node1:/usr/sbin/rsct/samples/tec # lssamctrl
```

The following IBM Tivoli System Automation control information is displayed:

```
SAMControl:
        TimeOut               = 60
        RetryCount            = 3
        Automation            = Auto
        ExcludedNodes         = {}
        ResourceRestartTimeOut = 5
        ActiveVersion         = [1.2.0.0,Fri Apr 16 16:05:50 2004]
        EnablePublisher       = Disabled
```

To enable the publisher issue this command on the master node:

```
node1:/usr/sbin/rsct/samples/tec # samctrl –e P
```

To disable the publisher issue this command on the master node:

```
node1:/usr/sbin/rsct/samples/tec # samctrl –d P
```

## Setting a new language locale for the TEC event messages

TEC event messages are always in the language which is the default system locale on the node where the IBM Tivoli System Automation for Multiplatforms master is running.

**Note:** Resource names in TEC event messages can be corrupted, if the user created the resources (mkrg, mkrsrc) in a shell with a different locale than the default system locale, or the terminal program has a different character set translation defined than the shell locale. To solve this problem, the system and shell locales must have identical settings and the character translation of the terminal program must be set accordingly. If the shell locale changes and resources are already created with the old shell locale setting, all resources must be deleted and have to be recreated with the new shell locale.

If the user chooses to adjust the default system locale to his preferred shell settings, then this change has to be done on all nodes of the cluster. Do the following to perform this:

1. Stop the cluster using the stoprpdomain command.
2. Edit the file containing the default system locale, set the appropriate values, and save the file.

   **SUSE Linux**

   File: /etc/sysconfig/language
   Keywords: RC_LANG="<NewLocale>"
   (replace <NewLocale> with your locale setting)
   ROOT_USES_LANG="yes"
   All keywords starting with RC_LC_ must be set
   to empty strings ""
   e.g. RC_LC_ALL= ""

Run */etc/SUSEconfig* to apply the changes to your system.
You can also use the *yast2 sysconfig* system configuration tool to apply the changes.

**RedHat Linux**

File: /etc/sysconfig/i18n
Keywords: LANG="<NewLocale>"
(replace <NewLocale> with your locale setting)

**AIX**

File: /etc/environment
Keywords: LANG="<NewLocale>"
(replace <NewLocale> with your locale setting)

3. Reboot the system.
4. Repeat the steps on all nodes in the cluster.
5. Start the cluster using the startrpdomain command.

## Publishing IBM Tivoli System Automation internal attributes into the RSCT infrastructure

This function makes IBM Tivoli System Automation internal attributes known to the RSCT infrastructure. For this purpose, the resource classes IBM.ResourceGroup, IBM.Equivalency, and IBM.ManagedResource are extended with the dynamic data structure attribute AutomationDetails. The dynamic data structure AutomationDetails has the following attributes:

- CompoundState – overall status of the resource including group dependencies. Shows how far the resource has reached the DesiredState. For example, ″Satisfactory″ means that the resource or resource group has reached the requested user status.
- DesiredState – user requested status of the resource. For example, ″online″ means that the user requested that the resource should be online.
- ObservedState – actual status of the resource from an automation perspective. For example, ″online″ means that the resource is currently online.
- BindingState – status indicating if the resource is bound to a specific system. For example, ″bound″ means that the resource is currently bound to a specific system.
- AutomationState – status indicating if the resource is currently being automated. For example, ″Idle″ means that IBM Tivoli System Automation is currently not trying to start or stop the resource.
- ControlState – status indicating if the resource can be controlled by automation. For example, ″startable″ means that it its currently possible to start this resource.
- HealthState – health status of the resource. This is reserved for future releases.

The **lsequ** and **lsrg** commands have been extended to show these attributes.

Each change of the value of one of these attributes indicates a status change of a resource and will be published to RSCT. If the TEC publisher is enabled (see “Enabling the TEC publisher function” on page 188), these status changes are also shown as TEC events.

# Enabling IBM Tivoli System Automation for GDPS/PPRC Multiplatform Resiliency for zSeries

Today, businesses and companies depend on disaster recovery solutions to recover critical data. This is why IBM Tivoli System Automation supports GDPS/PPRC Multiplatform Resiliency on System z (xDR).

Geographically Dispersed Parallel Sysplex (GDPS) is an application availability and disaster recovery solution which is highly customized to work with your z/OS environment. It provides disaster and failure recovery from a single point of control and ensures data consistency. For more information about GDPS, see the IBM Redbook *GDPS Family - An Introduction to Concepts and Capabilities, SG24-6374-01*, which can be downloaded at `http://ibm.com/redbooks`.

IBM Tivoli System Automation allows you to extend GDPS/PPRC for Linux systems running on System z. It provides a coordinated disaster recovery solution for systems running on zSeries, including z/OS, Linux on System z under z/VM, and Linux on System z running native in the LPAR.

**Notes:**

1. If you want to use the xDR functionality, particular versions of z/VM, Linux on System z, GDPS, and IBM Tivoli System Automation must be installed. For detailed information about the available functionality and the required versions, refer to the GDPS manuals. IBM Tivoli System Automation supports xDR for Linux on System z only.
2. The xDR naming conventions require that the names of clusters and nodes must not exceed 32 characters. For xDR, cluster names are not case-sensitive. To use xDR, IBM Tivoli System Automation must be customized as described in the GDPS manuals.
3. English is the only language supported by xDR and GDPS.

## Supported GDPS version

xDR requires GDPS V3.3 with APAR PK30315 or higher

## Supported Linux distributions

The following Linux distributions are supported for xDR:

- xDR for Linux on z/VM requires SUSE SLES 9 SP2
- xDR for Linux running native in LPAR requires SUSE SLES 10

## Dynamically verifying resources

Usually resource verifications are performed at configuration time when a resource is defined to IBM Tivoli System Automation. Then the user immediately gets notified when a problem occurs, and the definition of a new resource fails.

This may, however, not be the case after a resource has been defined and then a configuration change occurs. IBM Tivoli System Automation gets notified after the configuration change has been made, and then it has to accept and react to these changes, which eventually may lead to one or more resources becoming invalid.

This may happen when you, for example, define a resource using the mkrsrc command, change the values of a resource using the chrsrc command, or remove a defined resource using the rmrsrc command.

In order to verify such configuration changes and to convey the validity of a resource to the user, the resource classes IBM.ResourceGroup, IBM.ManagedResource, IBM.Equivalency, and IBM.ManagedRelationship contain a dynamic attribute ConfigValidity. ConfigValidity contains a string which explains why the resource is invalid.

Use the **lsrsrc –Ad** command to display the value of ConfigValidity together with the values of the other dynamic attributes of a resource.

The following verifications are performed:

- **A resource group's AllowedNode attribute is empty**

  When a node is removed, it may cause an equivalency to contain an empty member list. A resource group will become invalid if it uses this empty equivalency as its ″AllowedNode″ attribute. When this happens, the ″ConfigValidity″ dynamic attribute of the resource group will contain string "Allowed node list is empty".
- **Intersection of nested resource group's AllowedNode is empty**

  In a collocated resource group, all nested inner resource groups and the containing resource group must have at least one node in common. If there is only one node in common, and that node is removed, all the resource groups become invalid. When this happens, the ″ConfigValidity″ dynamic attribute of the resource group will contain the string ″No common node in collocated nested resource group″.
- **No node to run a resource**

  In a resource group, there may be the case when there is only one node in common between the resource group's AllowedNode and a member resource's NodeNameList. If this node is removed, the resource group becomes invalid. When this happens, the ″ConfigValidity″ dynamic attribute of the resource group will contain string ″No common node to start a resource″.
- **No node to satisfy a relationship**

  In a DependsOn relationship, with implied collocation, the NodeNameList of the source and NodeNameList of the target resource must have at least one node in common. If this node is removed, the relationship becomes invalid. When this happens, the ″ConfigValidity″ dynamic attribute of the managed relationship will contain string ″No common node between source and target″.
- **Cannot satisfy an AntiCollocated relationship - 1**

  If two mandatory floating resources in a resource group have an AntiCollocated relationship among each other, and removal of nodes leaves only one node in

the resource group's AllowedNode, the resource group becomes invalid. When this happens, the "ConfigValidity" dynamic attributes of the resource group and the AntiCollocation managed relationship will contain string "An AntiCollocated relationship cannot be satisfied".

- **Cannot satisfy an AntiCollocated relationship - 2**

  A removal of nodes causes two floating resources to have only one constituent left on the same node. But the two have an AntiCollocated relationship. When this happens, the "ConfigValidity" dynamic attributes of the AntiCollocated managed relationship will contain string "An AntiCollocated relationship cannot be satisfied".

- **Propagated invalidity**

  Any invalid inner resource group will cause all enclosing resource groups to become invalid. When this happens, the "ConfigValidity" dynamic attribute of the affected resource group will contain string "An enclosed resource group is invalid."

# IBM Tivoli System Automation Hints and Tips

This section provides various hints and tips which are helpful when operating a cluster with IBM Tivoli System Automation for Linux.

## Rebooting a node

Do not reboot a node when any resources are running on this node. First stop all running resources, using the following command:

```
samctrl -u a <node_name>
```

Now you can safely reboot the node.

This command also prevents resources from being started on this node. Reenable the node for running resources by entering:

```
samctrl -u d <node_name>
```

## Stopping a node

Before you stop a node (e.g. with he RSCT command **stoprpnode**), you must exclude this node from automation with the following command:

```
samctrl -u a <node_name>
```

The exclude must be done even if there are currently no resources online on the node.

After you have started the node again and it is online, you must reestablish automation on that node by entering:

```
samctrl -u d <node_name>
```

## Generating events in case of failures

RSCT has the ability to generate events in case a dynamic attribute of a resource changes. This is done by the event response resource manager (ERRM). The event response resource manager provides a set of commands that enable you to monitor events of interest (called conditions) and have the RMC system react in particular ways (called responses) if the event occurs.

So, for example, you can subscribe for the OpState of a resource group and get an email if the status changes. You can also monitor different resources in your system which are critical. For further explanation how to generate such events, see the chapter on basic resource monitoring in the *IBM Reliable Scalable Cluster Technology for Linux, Technical Reference*, SA22–7893.

# Chapter 12. Resource managers provided by IBM Tivoli System Automation

This chapter describes the resource managers provided by IBM Tivoli System Automation.

## Using the Global Resource Manager

This section describes the characteristics of the Global Resource RM.

The Global Resource RM (IBM.GblResRM) provides the following two resource classes:

1. IBM.Application:

   This class allows additional types of resources (e.g. business applications) to be monitored and controlled through the RMC subsystem. These resources can then be automated or recovered by management applications such as IBM Tivoli System Automation.
2. IBM.ServiceIP:

   This class is used to manage IP addresses that can be started, stopped, and moved between network adapters and nodes within a peer domain under the control of the RMC subsystem. These IP addresses will typically be provided to clients that are connecting to some service that is running within the domain. IBM Tivoli System Automation can be used to keep the service and its associated IP address active, even through failures within the domain.

The resource manager (and access to its classes) is operable in peer domain mode only.

The following subsections describe the external characteristics of the resource classes that are supported by this resource manager. Each of the subsections will describe one resource class, including its persistent and dynamic attributes, actions, etc.

### What is the IBM.Application resource class?

The IBM.Application resource class allows new types of floating and fixed resources to be created, monitored and controlled through the RMC subsystem. These resources can then be automated or recovered by IBM Tivoli System Automation. In order to create a new resource, the following three scripts (resp. commands) must be provided:

1. A start script (or command) to bring the resource online.
2. A stop script (or command) to take the resource offline.
3. A script (or command) to monitor the resource through polling.

Besides these scripts, there are the following basic parameters to the IBM.Application resource class:

1. The name of the resource.
2. The nodes where the resource can run.
3. A user name used to start/stop/monitor the application.

4. Method to be used to synchronously or asynchronously start/stop the application.
5. Different timeouts.
6. Characterization of the resource as either critical or non-critical.
7. Determination of the monitoring frequency.
8. Identification of the resource as fixed or floating.

Each generic resource that is instrumented through the IBM.Application resource class is considered to be a global resource meaning that it is not tied to a single node. However, the resource may be defined to exist on only a subset of the nodes of the cluster. For each generic resource, one instance of the IBM.Application resource class must be created. This instance is called an aggregate resource since it represents the floating resource that can move between nodes. In addition, there will be one instance of the IBM.Application resource class for each node where the generic resource exists. These are called constituent resources of the aggregate resource. Constituent resources are fixed resources in the sense that they exist on exactly one node of the cluster. Figure 23 illustrates the difference between aggregate and constituent resources.

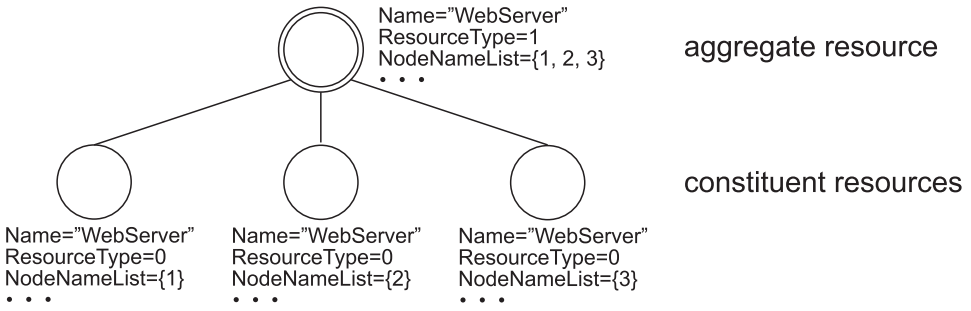



*Figure 23. Aggregate resource and constituent resources*

The constituent resources are automatically created or deleted as the definition of the aggregate resource is changed. Most management operations are done through the aggregate resource, but some applications may choose to monitor or operate on the constituents directly. Changes made to the aggregate resource are automatically applied to all constituents, whereas the change of an attribute of a constituent affects this constituent only and is not delivered to other resources (for example, the constituent on one node may have a different start command or monitoring interval).

When nodes are removed from the NodeNameList of the aggregate resource, the constituents are deleted automatically.

## Attributes used by IBM.Application

This section describes the attributes that are used by resources of the IBM.Application resource class.

When a resource of this class is created with the RMC command **mkrsrc**, it must have these attributes:

- Name
- StartCommand
- StopCommand
- MonitorCommand

- UserName

Resources of this class may have these attributes:

- NodeNameList
- ResourceType
- StartCommandTimeout
- StopCommandTimeout
- MonitorCommandTimeout
- MonitorCommandPeriod
- RunCommandsSync
- ProtectionMode

Resources of this class have the following dynamic attribute:

- OpState

**Name attribute**

The Name persistent attribute is a user– defined name for the generic application resource. Both the aggregate and constituent resource will have the same value for this attribute.
A value for this attribute must be specified when a new IBM.Application resource is created and it must be unique.
The attribute must be of type character string.

**NodeNameList attribute**

The NodeNameList persistent attribute is an array of strings that indicates which nodes the IBM.Application resource is available on. If the resource is floating, the Global Resource RM will ensure that there is one constituent (i.e. fixed) resource for each node name in this list. Constituent resources are implicitly created or deleted as necessary to match the entries in this list. Constituent resources will only contain one entry in this array since they are fixed resources and thus only available on one node. This attribute for a floating resource is implicitly modified if a constituent resource is explicitly removed by the administrator so that the aggregate and constituent resource relationship is always consistent.
This list may be empty for a floating resource which means that it is not available anywhere and constituents may be added separately.
The list may contain at most one name if the resource is fixed (i.e. ResourceType=0). If no name is given for a fixed resource, then RMC will provide a default because the fixed resource is tied to a node and therefore cannot be created without a node name or node id.
The value of this attribute for aggregate resources may be changed with the chrsrc command. An attempt to modify this attribute for a constituent resource will generate an error.

**ResourceType attribute**

With the ResourceType persistent attribute you identify whether the resource is fixed or floating. An integer value of 0 indicates that the resource is fixed, a value of 1 indicates that it is a floating resource. This attribute defaults to floating if not specified when a new IBM.Application resource is created.

**StartCommand attribute**

The value of the StartCommand persistent attribute contains the

exact command and arguments that will be executed when the resource manager receives a start request for the corresponding resource instance. The command is only executed by constituent resources even if the online request was issued to the aggregate resource. In this case, the resource manager will choose a constituent resource if not specified to execute the online request.
The command is executed under the user id specified with the UserName attribute.
The command is run with the authority and environment of the specified user.
Whether the resource manager waits for the command to complete is controlled by the RunCommandsSync attribute (see below for details). The command name must be a character string and it must be an absolute path (i.e. it must begin with a '/'). It must exist and be executable on each node where the resource is accessible (i.e. where there is a constituent).
This attribute must be specified when a new IBM.Application resource is defined.
The command may return the following values:

**0** Command has run successfully.

**!= 0** Error occurred during command processing.

See "How the resource manager handles return codes of the StartCommand, StopCommand, and MonitorCommand" on page 205.

**StopCommand attribute**

The value of the StopCommand persistent attribute contains the exact command and arguments that will be executed when the resource manager receives a stop request for the resource instance. A stop request for the aggregate will be issued by all constituents. All other aspects related to running the command are the same as for StartCommand.
This attribute must be specified when a new IBM.Application resource is defined.
The command may return the following values:

**0** Command has run successfully.

**!= 0** Error occurred during command processing.

See "How the resource manager handles return codes of the StartCommand, StopCommand, and MonitorCommand" on page 205.

**MonitorCommand attribute**

The value of the MonitorCommand persistent attribute contains the exact command and arguments that will be executed periodically to determine or update the operational state (OpState attribute) of the resource. The exit value from the command is used as the new OpState of the resource:

Unknown=0
Online=1
Offline=2
Failed Offline=3
Stuck Online=4
Pending Online=5

Pending Offline=6

At least the Online and Offline status should be set by the MonitorCommand script.
The IBM.GblResRM runs this command every MonitorCommandPeriod seconds when there are any subscribers to the OpState dynamic attribute. All other aspects of running the command are identical to those described under StartCommand. To avoid consuming system resources, this command should be as efficient as possible.
The command should return within a reasonable amount of time, usually within a few seconds, at least within 360 seconds.
The name of the MonitorCommand must be an absolute path (i.e. it must begin with a '/'). It must exist and be executable on each node where the resource is accessible (i.e. where there is a constituent).
This attribute must be specified when a new IBM.Application resource is defined. To learn more about the return value of the MonitorCommand attribute see "Important issues when defining IBM.Application resources" on page 204 and "How the resource manager handles return codes of the StartCommand, StopCommand, and MonitorCommand" on page 205.

**MonitorCommandPeriod attribute**

The value of the MonitorCommandPeriod persistent attribute specifies the amount of time (number of seconds) to wait between invocations of the MonitorCommand. This period is started after the prior invocation completes. This attribute must be of type integer and it must be greater than 0. It defaults to 5 seconds.

Since the period starts after the prior invocation completes, the value of the MonitorCommandTimeout attribute can be greater than the value of the MonitorCommandPeriod attribute.

**MonitorCommandTimeout attribute**

With the MonitorCommandTimeout persistent attribute you specify the amount of time a monitor command is allowed to run before it is killed via killpg(). If the command times out, the operational state (OpState attribute) of the resource is set to Unknown=0.
This attribute must be of type integer and it must be greater than or equal to 0.
The value must be lower than 360 seconds. The default value for this attribute is 5 seconds.

**StartCommandTimeout attribute**

With the StartCommandTimeout persistent attribute you specify the amount of time a start command is allowed to run before it is killed via killpg(). Furthermore, this attribute also specifies the amount of time after which IBM Tivoli System Automation expects the resource to be online, i.e.IBM Tivoli System Automation uses this value instead of the default timeout given with the control parameters.
This attribute must be of type integer and it must be greater than or equal to 0. A value of 0 for this attribute means no timeout. The attribute is not used if the RunCommandSync attribute is set to 0.
The default value for this attribute is 5 seconds.

**StopCommandTimeout attribute**

With the StopCommandTimeout persistent attribute you specify the

amount of time a stop command is allowed to run before it is killed via killpg().This attribute must be of type integer and it must be greater than or equal to 0. A value of zero means no timeout. The default value for this attribute is 5 seconds.

**RunCommandsSync attribute**

You use the RunCommandsSync persistent attribute to control whether the start/stop commands are executed synchronously with the online()/offline() method. If the value of this attribute is set to the integer value 1 which is the default, then the response to the online()/offline() methods will not be completed until the command completes or times out. Any stderr/stdout outputs will be returned in the response for this case. If the value of this attribute is 0, then the IBM.GblResRM will ″fire and forget″ the start/stop commands. As soon as the fork/exec completes successfully, the resource manager forgets about them and they run completely unattached from the resource manager.
This attribute defaults to 1. Timeouts will not be applied to commands when this attribute is set to 0.
If your StartCommand is the application executable and the command does not return in a certain amount of time, but is running as long as your application is running, you cannot use the synchronous mode. In case of the synchronous mode the resource manager would kill the command after the StartCommand timeout has expired.

Use RunCommandsSync=1 when you know how long the StartCommand normally takes to complete. Adapt the StartCommandTimeout attribute to this time. During heavy system load this time may be longer even in case of a normal start up without any error conditions.

Use RunCommandsSync=0 when the command does not complete until your application is running (e.g. the executable of your application). If you want to run the application executable directly (asynchronous mode), but use the synchronous command mode and send the command to the shells background, redirect I/O file descriptors. If the resource manager is still connected to the process, I/O descriptors the StartCommand timeout will kill your process.

The RunCommandsSync attribute also controls the environment the start/stop and monitor command is executed in. You can either select a basic environment for the commands or the resource manager can run a full-blown login environment including the user profile and shell configuration files. You can compare this to the system command 'su' (switch user). Either you keep the current environment or you can do a 'su –' and run the full profile (login shell) for the selected user. Following values are allowed for the RunCommandsSync resource attribute:

| Value | Description |
|---|---|
| 0 | Run commands in asynchronous mode.<br>Run commands in a limited environment. |
| 1 (default) | Run commands in synchronous mode.<br>Run commands in a limited environment. |

| Value | Description |
|---|---|
| 2 | Run commands in asynchronous mode.<br>Run profile and login scripts for the specified user and build the users login environment. |
| 3 | Run commands in synchronous mode.<br>Run profile and login scripts for the specified user and build the users login environment. |

Based on the operating system (AIX or Linux) the limited environment for the user contains following environment variables (values of the variables represent a sample user):

```
SHELL=/bin/bash
USER=myuser
PATH=:/usr/ucb:/bin:/usr/bin
LOGIN=myuser
PWD=/home/myuser
LANG=de_DE.UTF-8
HOME=/home/myuser
SHLVL=2
LOGNAME=myuser
```

If this is not sufficient for the specified start/stop and monitor commands, either set the necessary variables in the start/stop or monitor command itself, or put them into the users profile (or shell configuration) and select the login environment for the commands (RunCommandsSync=2|3).

**UserName attribute**

The UserName persistent attribute defines a user name under which the MonitorCommand, StartCommand and StopCommand are run. The commands are run with the authority and environment of the specified user. A check will be made on each node to ensure that the user name exists whenever the configuration of the resource is modified.
This attribute must be specified when a new IBM.Application resource is defined. The attribute must be of type character string.

**ProtectionMode attribute**

You use the ProtectionMode persistent attribute to specify whether the resource is critical. If it is critical, then the IBM.ConfigRM decides if the resource can be started as requested. (For further detail on this behavior, see Chapter 9, “Protecting your resources – quorum support,” on page 137).
The attribute may have the integer values 0 (Non-Critical) or 1 (Critical), it defaults to Non-critical. If the resource is set to Critical, monitoring will immediately start, even if no subscriber to this resource exists.

**OpState attribute**

The value of this dynamic state attribute contains the operational state of the resource as determined by the exit code from running the MonitorCommand periodically. The possible values for this attribute as defined by the state transition diagram in the RMC architecture are:

Unknown=0

Online=1

Offline=2
Failed Offline=3
Stuck Online=4
Pending Online=5
Pending Offline=6

This attribute is available from both the aggregate resource and all constituent resources. The value for an aggregate resource is a roll-up of the states from each of the constituents.

**HealthCommand**
Reserved for future use.

**HealthCommandPeriod**
Reserved for future use

**HealthCommandTimeout**
Reserved for future use

**InstanceName**
Reserved for future use

**InstanceLocation**
Reserved for future use

## Actions used by IBM.Application

This section describes the actions that can be performed on resources of the IBM.Application resource class.

**The refreshOpState action**
In normal operation the MonitorCommandPeriod attribute determines the interval, the OpState of an IBM.Application resource is evaluated by its monitoring script. In case a resource is able to detect a failure by itself, it is possible to trigger an immediate run of the monitoring script which is monitoring the resource. This will result in an immediate OpState refresh of the application resource.

For example, to refresh the OpState of the resource 'WebServer' running on node02, issue the following command on any node:

```
runact -s "Name='WebServer' && NodeNameList={'node02'}" IBM.Application refreshOpState
```

**SendEIFevent** Reserved for future use.

## Important issues when defining IBM.Application resources

Keep the following in mind when defining an IBM.Application resource:

1. In order to satisfy goal driven automation IBM Tivoli System Automation resources are always monitored. No matter if the resource is started or stopped, the monitor command has to determine the actual state of the resource. Do not put the monitor command on a file system which is not always present (for example, a NFS mount which is part of a policy and is only mounted if the NFS resource is started). If you see an IBM.Application OpState of "unknown", check the system log and take care the GblResRM resource manager can access the monitor command at any time.
2. The GblResRM resource manager will kill any command which is running longer than the timeout value allows. If you see IBM.Application resources reporting OpState "unknown" during heavy system load, the command probably takes longer to complete than the timeout value of this resource allows. If there is more CPU time, this will not lead to problems.The monitor

command will be able to complete and report a valid OpState again. Check the system log to see if the command was killed cause of the timeout. If commands are killed during normal operation, check and adjust the timeout values in your resource definition.

3. The monitor must clearly identify the process or application it is responsible for. If the monitor, for instance looks into the process table for a particular process, this process must be uniquely identified. If the monitor monitors another process than automation actually controls, the behavior of the whole resource would be somehow unpredictable.
4. If you try to automate base system services like printer spooler or mail transport agent, take care this services are only started/stopped by the automation and not by the system runlevel or init process.
5. If you have a more sophisticated monitor which could not only determine if a process exists but also if the application is able to deliver service, there is no possibility to report this fact to the automation engine and trigger a stop or shutdown of this application. If the monitor finds out that the process or application is available but hang or stuck, the monitor itself should kill this process/application. Automation will recognize the application is not available anymore (next monitoring) and will try to restart the application in place or move it to another node.
6. If you try to remove an online resource of class IBM.Application (e.g. with rmrsrc), the command will be rejected. You can force removal by setting Force=1. For example, to remove a resource called WebServer you would enter:

   ```
   rmrsrc -s "Name=='WebServer' && ResourceType==1" IBM.Application Force=1
   ```
7. The soft limit for the number of open files is 1024. The hard limit is at the maximum, so the soft limit can be adjusted if needed within the scripts executed as StartCommand, StopCommand, or MonitorCommand for a resource.

   To increase the limit, add the following statement below the header of the *Command scripts for the resource:

   ```
   # ulimit -S -n <new-limit-for-number-of-open-files>
   ```

   This only has an impact on applications requiring a huge number of open files, like, for example, WebSphere Application Server.

## How the resource manager handles return codes of the StartCommand, StopCommand, and MonitorCommand

The resource manager handles the return codes for the StartCommand, StopCommand, and MonitorCommand as follows:

**StartCommand:**

1. If the StartCommand was able to start the resource it should return a value of 0 to indicate that the resource was properly started and should go online within the next few seconds.
2. If the StartCommand was not able to start the resource it should return a value other than 0. This signals the automation not to start the resource again and set the resource operational state to Failed Offline. This indicates that you have to manually intervene to fix the resource. When then the resource is able to start, reset the Failed Offline operational state with the **resetrsrc** command. Note that whenever a StartCommand failed, automation will issue a StopCommand to ensure application leftovers of the failing start are removed.
3. If the StartCommand completes successfully and returns a value of 0, but the MonitorCommand of the resource does not report an online state after a certain amount of time (depending on the settings in the automation control

configuration), the automation will try to restart the resource. The total number of start attempts is defined in the RetryCount attribute (see the description of the **lssamctrl** command in the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*), the default value is 3. If the operational state of the resource does not change to Online after the last attempt, a StopCommand is issued and the resource is set to Failed Offline.

4. If the StartCommand of a resource is not able to complete within the time specified in the StartCommandTimeout attribute, the resource manager will kill the StartCommand and treat the start like a failing start command as described under item 2 on page 205.
5. In case the StartCommand was valid when the resource was defined, but is later removed or not present (for example, because of a missing NFS mount), the start procedure is treated like a failing StartCommand as described under item 2 on page 205.

**StopCommand:**

1. If the StopCommand was able to stop the resource it should return a value of 0 to indicate that the resource was properly stopped and should go offline within the next few seconds.
2. There is no mechanism to handle a failed StopCommand in IBM Tivoli System Automation for Multiplatforms. The StopCommand may indicate a failing stop of the application by returning a value other than 0, but this will not trigger an automation action.

   If the resource does not reach the OpState Offline after a certain amount of time, IBM Tivoli System Automation for Multiplatforms issues a reset operation against the resource, which results in a second invocation of the StopCommand. This second invocation of the StopCommand can be determined within the StopCommand script by checking the SA_RESET environment variable, which is set to 1 for the second invocation. A StopCommand can honor this, like in the following example:

   ```
   #/bin/sh
   # A sample stop/reset automation script for the lpd application
   if [ $SA_RESET == 1 ]; then
     killall -9 lpd
     exit $?
   else
     /etc/init.d/lpd stop
     exit $?
   fi
   ```

   If a second execution of the StopCommand is not desired, the script could just exit, for example:

   ```
   #/bin/sh
   # A sample stop/reset automation script for the lpd application
   if [ $SA_RESET == 1 ]; then
     exit 0
   else
     /etc/init.d/lpd stop
     exit $?
   fi
   ```

   This feature is introduced in IBM Tivoli System Automation for Multiplatforms, release 2.2. However, rewriting existing policy scripts is not required as they will work as before. But by evaluating the SA_RESET environment variable, an enhanced stop behavior can now be easily achieved. If the second StopCommand also fails to bring the operational state of the resource to Offline, the OpState of the resource is set to Stuck Online. At this time, operator invention is required to stop the resource. After the resource has finally stopped

and the OpState has changed to Offline, IBM Tivoli System Automation for Multiplatforms will again control this resource.

3. If the StopCommand of a resource is not able to complete within the StopCommandTimeout, the resource manager will kill the command and treat the stop like a failing StopCommand as described under item 2 on page 206.
4. In case the stop command was valid when the resource was defined, but is later removed or not present (for example, because of a missing NFS mount), the stop procedure is treated like a failing StopCommand as described under item 2 on page 206.

**MonitorCommand:**

1. If the MonitorCommand was able to determine the operational state of a resource it should return one of the valid RMC operational states (see page 203). Keep in mind that in this case 0 is not the return value for RMC operational state Online, but the return value for operational state Unknown, which is the most critical state for the automation. A resource with an unknown operational state will not be automated anymore, and this may also affect other resources which have dependencies to this resource.
2. If the monitor command of a resource is not able to complete within the MonitorCommandTimeout, the resource manager will kill the MonitorCommand and set the RMC operational state to Unknown, which indicates a major problem with the resource. There will be no automation with this resource, until the MonitorCommand returns an operational state other than Unknown.
3. In case the monitor command was valid when the resource was defined but is later removed or not present (for example, because of a missing NFS mount), the operational state is set to Unknown, which indicates a major problem with the resource.
4. In both cases the MonitorCommand may continue to report valid RMC operational states after system load has decreased or NFS is present again, and now automation will continue automating the resource.
5. During the execution of the StartCommand of a resource, the OpState of the resource will always be set to Pending Online, regardless of the actual OpState reported by a MonitorCommand, to reflect that the resource is actually in the process of starting. There are two exceptions - if the MonitorCommand returns Online or Failed Offline – because these OpState values indicate that the resource is already online or broken and cannot be started.
6. During the execution of the StopCommand of a resource, the OpState of the resource will always be set to Pending Offline, regardless of the actual OpState reported by a MonitorCommand, to reflect that the resource is actually in the process of stopping. There are three exceptions - if the MonitorCommand returns Offline, Failed Offline, or Stuck Online – because these OpState values indicate that the resource is already offline or stuck and cannot be stopped.

## How the Global Resource Manager creates processes for the StartCommand, StopCommand, and MonitorCommand of IBM.Application

The first process in the UNIX/Linux kernel is the *init* process which creates and starts (spawns) the system resource controller (src). As shown in the following figure, the system resource controller is responsible for the IBM Tivoli System Automation resource managers.

```
Init-+-atd
     |
     |-srcmstr
     |          |
     |          |-IBM.GblResRMd---IBM.GblResRMd---13*[Ibm.GblResRMd]
```

The Global Resource Manager creates additional processes to run the start/stop and monitor commands of an IBM.Application resource. All commands created by the Global Resource Manager run in the shell of the specified user. Here is some pseudo code of this functionality:

```
{
  fork;
  if child
    switch to specified user ID;
    run the users default shell and execute the command e.g.
      bash -c /usr/bin/mycommand;
  endif
}
```

The command itself, such as mycommand in the previous shown pesudo code, can be any executable including shell scripts which may create additional processes or use job control.

The following describes various scenarios:

1. IBM.Application is defined in asynchronous command mode:

   ```
   StartCommand="/usr/bin/mycommand"
   RunCommandsSync=0
   ```

   In this case the Global Resource Manager creates (forks) a process for the command, then it completely detaches from the new process and closes all file descriptors to the new process.

   The Global Resource Manager does not take care of the new process anymore, and in theory the command could run forever.

   Since the new process does not have a parent anymore, it becomes an orphan and is adopted by the init process. When the process eventually ends, init will collect the return code of the process.

2. IBM.Application is defined in synchronous command mode:

   ```
   StartCommand="/usr/bin/mycommand"
   RunCommandsSync=1
   ```

   Here the Global Resource Manager does not detach itself from the newly created (forked) process. File descriptors are open, and the resource manager waits for completion of the command:

   - If the command returns a bad return code, messages from stderr are captured and written to the Global Resource Manager trace and error block of the start/stop command.
   - If the command does not return within the time specified in the StartCommandTimeout attribute, the Global Resource Manager sends SIGKILL to the forked process (the users default shell). SIGKILL is propagated to all child processes of the user shell and therefore all child processes should end.

3. IBM.Application is defined in synchronous command mode, but uses job control of the user shell:

   ```
   StartCommand="/usr/bin/mycommand &"
   RunCommandsSync=1
   ```

In this setup the user default shell will not end until all child processes which have open file descriptors to the shell have ended. Here the time specified in the StartCommandTimeout attribute also applies to mycommand, even it is run in background of the user shell. So if mycommand runs longer as the time specified in the StartCommandTimeout attribute, the Global Resource Manager will send SIGKILL to the user shell and it will propagate SIGKILL to all background process.

If you want to make this setup working, you must make sure that all file descriptors of the shell child process are detached from the shell. This may look like:

```
StartCommand="/usr/bin/mycommand > /dev/null 2>&1 &"
RunCommandsSync=1
```

Now the user shell can end right after it has created (forked) the process for the command and has detached from the file descriptors of the new process. The command (mycommand) itself behaves as described in the first scenario, it becomes an orphan and is adopted by the init process..

## Example: Implement the lpd printer spooler as an IBM.Application resource

The following example shows how to prepare the lpd printer spooler on a SUSe based Linux system to be managed by IBM Tivoli System Automation.

1. Remove the lpd from the default runlevel of the system. If you want to run this resource as a floating resource on more than one node, you have to check the runlevel on each node.
2. For the start and stop command of the IBM.Application use the default init scripts shipped with your lp daemon:

   ```
   StartCommand: /etc/init.d/lp start
   StopCommand: /etc/init.d/lp stop
   ```

3. For the monitor command we use a simple shell script which checks for the lpd process in the process table:

   ```
   File: /root/lpmon
   #!/bin/bash

   OPSTATE_ONLINE=1
   OPSTATE_OFFLINE=2

   ps -ax | grep -v "grep" | grep "/usr/sbin/lpd" > /dev/null
   if [ $? == 0 ]
   then
     exit $OPSTATE_ONLINE
   else
     exit $OPSTATE_OFFLINE
   fi
   ```

   Alternatively you can use the **pidmon** command shipped with IBM Tivoli System Automation. It basically searches the process table for a given command string. If the command string was found, The RMC OpState is returned. See the *IBM Tivoli System Automation for Multiplatforms Base Component Reference* for a detailed description of this command.

   ```
   MonitorCommand: /root/lpmon
   or
   MonitorCommand: /usr/sbin/rsct/bin/pidmon '/usr/sbin/lpd'
   ```

4. In case of a floating resource make sure that all nodes can access the start/stop and monitor command under the same path. Since the lpd is a small and simple application, the default Start-/Stop- and MonitorCommandTimout

values (default is 5 seconds) can be used. In order to start lpd via the init scripts provide root as the user name for the IBM.Application.

Now the IBM.Application resource can be defined using the **mkrsrc** command:

```
# mkrsrc IBM.Application  \
        Name                 = "line_printer_daemon"  \
        ResourceType         = 1  \
        StartCommand         = "/etc/init.d/lpd start"  \
        StopCommand          = "/etc/init.d/lpd stop"  \
        MonitorCommand       = "/usr/sbin/rsct/bin/pidmon '/usr/sbin/lpd' "  \
        MonitorCommandPeriod  = 15  \
        MonitorCommandTimeout = 5  \
        StartCommandTimeout   = 5  \
        StopCommandTimeout    = 5  \
        UserName             = "root"  \
        RunCommandsSync      = 1  \
        ProtectionMode       = 0  \
        NodeNameList         = "{'node01','node02'}"
```

This command results in three resources being created: An aggregate resource named ″line_printer_daemon″ which can potentially be brought online on nodes ″node01″ and ″node02″ and two constituent resources also named ″line_printer_daemon″, one on node ″node01″ and the other on node ″node02″. If a start request is issued against the aggregate resource, then the Global Resource RM chooses one of the constituents and starts it with the script (or command) specified with the StartCommand attribute.

## Configuring a supporting resource for an IBM.Application resource

If you use an IBM.Application in combination with an IBM.Equivalency and DependsOn relationship, automation will choose a resource from the equivalency and provide this resource as a supporting resource to the IBM.Application.

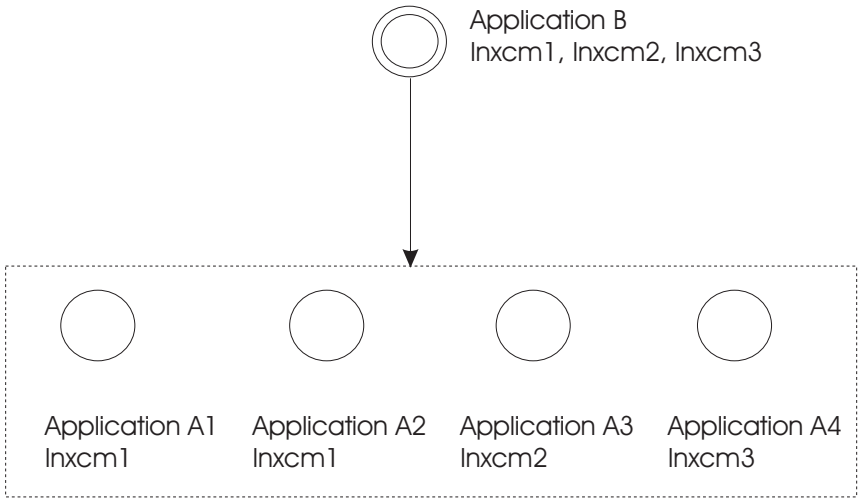



*Figure 24. Configuring a supporting resource for an IBM.Application resource*

In this sample configuration automation picks an application Ax from the equivalency and takes care application B will be depend on this application (see the chapters Chapter 5, “Using equivalencies,” on page 47 and Chapter 6, “Using managed relationships,” on page 53). Since more resources from the equivalency can fulfill the location constrains of the DependsOn relationship (in this sample A1

and A2 can run on lnxcm1), automation will provide the information which resource was selected to the start command of resource B.

If necessary the start script of resource B can use this information to pass special parameters to its application, or activate dedicated code which has to be carried out in combination with the selected resource from the equivalency.

To pass this information to the start command, the resource manager will set following environment variables in the start command environment of the resource:

| Variable | Value |
|---|---|
| SA_SUPPORTING_RESOURCE_RH | The resource handle of the supporting resource, such as "0x601d 0xffff 0xcac15160 0xbad91087 0x0f933128 0x58888f98" |
| SA_SUPPORTING_RESOURCE_NAME | The name of the supporting resource, such as "A1" |

Here is some sample code which shows how the start script for resource B can honor the supporting resource environment variables:

```
# start command for resource B with logic for
# supporting resource
...
if [ $SA_SUPPORTING_RESOURCE_NAME = "A1" ]
then
  # start resource B and connect it to supporting
  # resource A1
  ...
fi

if [ $SA_SUPPORTING_RESOURCE_NAME = "A2" ]
  # start resource B and connect it to supporting
  # resource A2
  ...
then
fi
```

## What is the IBM.ServiceIP resource class?

The IBM.ServiceIP resource class is used to manage IP addresses that can be started, stopped and moved between adapters and nodes within a peer domain. Each resource of this class identifies one IP address. These IP addresses will typically be provided to clients that are connecting to some service that is running within the domain. A recovery management application can be used to keep the service and its associated IP address active, even through failures within the domain.

A Service IP address is an aggregate resource with one constituent resource per node where the administrator wants to allow that resource to be potentially brought on line. It is a floating aggregate resource since it can only have one constituent active at a time.

The IBM.ServiceIP resource class uses the following basic parameters:

1. The name of the resource.
2. The nodes where the resource can run.
3. The IP address which can be moved.
4. The netmask of the IP address.

The broadcast address and flag for the network interface will be taken from the parent interface where the ServiceIP resource is aliased on during the start.

Keep in mind that IBM.ServiceIP will generate a static routing entry for the network the IBM.ServiceIP is in. Take care that this network/route will not destroy the network configuration of the device the ServiceIP is aliased on.

Also automation does not take care of dynamic routing. If you specify a ServiceIP which is not in the subnet of the parent network interface, this physical device will host two different networks. Be sure to properly set up routing outside the automation cluster in order to support this network.

## Characteristics of IBM.ServiceIP

The IBM.Service IP class has two different flavours:

1. **IBM.ServiceIP automatically chooses a suitable network interface**

   If the IBM.ServiceIP receives a start request, the resource manager tries to choose a suitable network interface. If it can find one, the service IP is aliased on the that interface. This interface is called the supporting resource or supporting network interface.

   In order to determine a suitable network interface, the resource manager compares the IPAddress attribute of the IBM.ServiceIP resource to the IP address of all existing network interfaces. If there is a suitable network (subnet) match found, the resource manager assigns the alias address to that network interface. If no matching subnet is found, the online request fails.

   In case of the automatic interface choosing algorithm automation has no chance to evaluate the actual state of the network interface. As long as the network interface is configured and running, the ServiceIP can be assigned to the network interface. Keep in mind that UNIX/Linux do not change the status of a network interface, for example if the network cable is unplugged. Even if the device driver is able to detect the missing cable, the interface will stay configured and running.

   If you want to exploit RSCT heartbeat mechanism to detect a network interface failure like a missing cable, use the supporting resource setup of the IBM.ServiceIP described in the next section.

   Example

   An IBM.ServiceIP is defined as:

   ```
   IPAddress=192.168.1.5
   NetMask="255.255.255.0"
   NodeNameList="{'node01','node02'}"
   ```

   The following network interfaces are available in the cluster:

   1. IPAddress=192.168.1.1
      Netmask=255.255.255.0
      NodeNameList="{'node01'}"

   2. IPAddress=9.152.172.91
      Netmask=255.255.255.0
      NodeNameList="{'node02'}"

   3. IPAddress=192.168.2.1
      Netmask=255.255.255.0
      NodeNameList="{'node03'}"

Only interface Number 1 is able to hold the service IP. All other interfaces do not match the address (subnet) of the service IP. Calls to start the IBM.ServiceIP on other nodes than node01 will fail, because there is no suitable supporting network interface.

2. **IBM.ServiceIP receives a supporting resource to alias the service IP address on**

   In this case there are no limitations for the IPAddress and the NetMask attribute of the IBM.ServiceIP resource. The service IP must have a DependsOn relationship (see "DependsOn relationship" on page 64) to an equivalency of network interfaces (see Chapter 5, "Using equivalencies," on page 47 for supporting resources). If IBM Tivoli System Automation decides to bring a service IP resource on a particular node online, it picks a suitable network interface from the equivalency of network interfaces and provides it to the IBM.ServiceIP as supporting resource. This is the interface hosting the alias.

   In this configuration automation is able to monitor the operational state of the network interface the service IP address is aliased on. In case of an interface failure detected by RSCT heartbeat, automation will force down the dependency chain and stop the ServiceIP. If there is another online network interface in the equivalency, automation will choose this device to assign the ServiceIP on.

   Example:

   An IBM.ServiceIP is defined as:

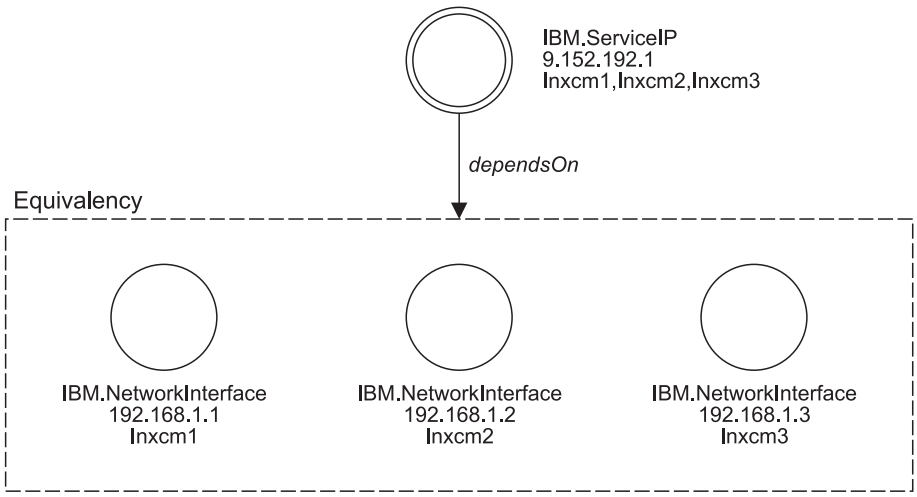
   ```
   IPAddress=9.152.192.1
   NetMask="255.255.255.0"
   NodeNameList="{'node01','node02','node03'}"
   ```

   The following network interfaces are available in the cluster:

   ```
   1. IPAddress=192.168.1.1
      Netmask=255.255.255.0
      NodeNameList="{'node01'}"
   2. IPAddress=192.168.1.2
      Netmask=255.255.255.0
      NodeNameList="{'node02'}"
   3. IPAddress=192.168.1.3
      Netmask=255.255.255.0
      NodeNameList="{'node03'}"
   ```

   All three network interfaces form an equivalency.

The following picture illustrates the setup:



If IBM Tivoli System Automation decides to start the service IP, it will pick a network interface from the equivalency. In this example the service IP can float between node01, node02 and node03, because all three nodes have a network interface in the equivalency which can act as supporting resource.

## Attributes used by IBM.ServiceIP

This section describes the attributes that are used by resources of the IBM.ServiceIP resource class.

When a resource of this class is created with the RMC command **mkrsrc**, it must have these attributes:

- Name
- IPAddress

Resources of this class may have these attributes:

- NodeNameList
- ResourceType
- NetMask
- ProtectionMode

Resources of this class have the following dynamic attribute:

- OpState

**Name attribute**

The Name persistent attribute is a user defined name for this service IP address (e.g. mail-server-ip). Both the aggregate and constituent resource will have the same value for this attribute.
A value for this attribute must be specified when a new IBM.ServiceIP resource is created and it must be unique.
The attribute must be of type character string.

**NodeNameList attribute**

The NodeNameList persistent attribute is an array of strings that indicates which nodes the IBM.ServiceIP resource is available on. If the resource is floating, the Global Resource RM will ensure that there is one constituent (fixed) resource for each node name in this list. Constituent resources are implicitly created or deleted as necessary to match the entries in this list. Constituent resources will only contain one entry in this array since they are fixed resources and thus only available on one node. This attribute for a floating resource is implicitly modified if a constituent resource is explicitly removed by the administrator so that the aggregate and constituent resource relationship is always consistent.
This list may be empty for a floating resource which means that it is not available anywhere and constituents may be added separately.
The list may contain at most one name if the resource is fixed (i.e. ResourceType=0). If no name is given for a fixed resource, then RMC will provide a default because the fixed resource is tied to a node and therefore cannot be created without a node name or node id.
The value of this attribute for aggregate resources may be changed with the chrsrc command. An attempt to modify this attribute for a constituent resource will generate an error.

**ResourceType attribute**

You use the ResourceType persistent attribute to identify whether the resource is fixed or floating. An integer value of 0 indicates that the resource is fixed, a value of 1 indicates that it is a floating resource. This attribute defaults to floating if not specified when a new IBM.ServiceIP resource is created.

**IPAddress attribute**

With the IPAddress persistent attribute you specify the IP address that will be aliased onto a network interface where the resource is brought online. This attribute is required when a new IBM.ServiceIP resource is created. The IP address must be given in 'dotted decimal' notation as a character string, e.g. 9.152.80.251.

**NetMask attribute**

You use the NetMask persistent attribute to specify the netmask that will be assigned to the IP address defined in the IP address attribute. The attribute must be given as a character string, for example 255.255.255.0.

**ProtectionMode attribute**

The ProtectionMode persistent attribute specifies whether the resource is critical. If it is critical, then the IBM.ConfigRM decides if the resource can be started as requested. (For further detail on this behavior, see Chapter 9, "Protecting your resources – quorum support," on page 137.

The attribute may have the integer values 0 (Non-Critical) or 1 (Critical), it defaults to Critical.

**OpState attribute**

The value of this dynamic state attribute contains the operational state of the resource as determined by the resource manager. Typical values for this state are Online (value is 1) and Offline (value is 2) meaning that the IP address is either operational or not operational.

## What happens when an IBM.ServiceIP is started?

If the resource manager is able to assign the IP address on the selected network interface, the following functions will be carried out:

1. On Linux on System z with OSA network hardware an IP address takeover will be initiated for the specified IP address.
2. The ServiceIP is created as an IP alias on the selected network device.
3. To invalidate ARP (Address Resolution Protocol) cache entries of other IP host which may have the ServiceIP with a wrong hardware address (MAC address) in their cache, an unsolicited/gratuitous ARP packet is broadcasted into the network.
4. There is a system log message created to record the fact a ServiceIP is started on the specified network interface.

Also note that

- An IP address alias may generate a new entry in the system routing table of the host (in case the ServiceIP network differ form the network of the device).
- The IBM.ServiceIP will not modify your default gateway settings or other network/routing configurations.

### Example 1: Define an IP address as an IBM.ServiceIP resource

In order to define an IP address which has the address IP 9.152.172.11, netmask 255.255.255.0 and potentially runs on the nodes node05 and node06, issue the following RMC command:

```
mkrsrc IBM.ServiceIP
  Name="WebServerIP"
  NodeNameList="{'node05','node06'}"
  IPAddress=9.152.172.11
  NetMask=255.255.255.0
```

### Example 2: Define an IP address as an IBM.ServiceIP resource and use an IBM.Equivalency of network interfaces

As shown in the preceding example, define an IP address which has the address IP 9.152.172.11, netmask 255.255.255.0 and potentially runs on the nodes node05 and node06:

```
mkrsrc IBM.ServiceIP
  Name="WebServerIP"
  NodeNameList="{'node05','node06'}"
  IPAddress=9.152.172.11
  NetMask=255.255.255.0
```

The nodes node05 and node06 have each more than one network interface. To form an equivalency containing eth1 device of node node05 and node node06, type:

```
mkequ MyInterfaces IBM.NetworkInterface:eth1:node05,:eth1:node06
```

Now you can connect the ServiceIP with the equivalency:

```
mkrel -p dependson -S IBM.ServiceIP:WebServerIP -G IBM.Equivalency:MyInterfaces
         WebIp_depon_MyInterfaces
```

# Using the Test Resource Manager

This section describes the characteristics of the Test resource manager.

The IBM Test resource manager (IBM.TestRM) manages test resources and provides functions to manipulate the operational state of these resources. The resource manager is operational in a peer domain mode only and provides the resource class IBM.Test. IBM.TestRM does not control real resources.

## What is the IBM.Test resource class?

The IBM.Test resource class allows new types of fixed and floating resources to be created, monitored and controlled through the RMC subsystem. These resources are no real resources but just the containers to define, monitor and control them. These resources can then be automated or recovered by IBM Tivoli System Automation. The purpose of the IBM.Test class is to provide a lightweight and easy to handle resource to simulate automation scenarios without the overhead of real resources. Each resource controlled by the IBM.Test class is considered to be a globalized resource which divides in one aggregate and one constituent on every node the resource is defined. See Figure 23 on page 198 for details.

The IBM.Test resource class provides a set of persistent resource attributes to simulate the behavior of real resources.

## Attributes used by IBM.Test

This section describes the persistent attributes that are used by the IBM.Test resource class.

When a resource of this class is created with the RMC command **mkrsrc**, it must have the following persistent attribute

- Name

Resources of this class may have these attributes:

- NodeNameList
- ResourceType
- ForceOpState
- TimeToStart
- TimeToStop
- WriteToSyslog
- MoveTime
- MoveFail

Resources of this class have the following dynamic attributes:

- OpState
- MoveState
- OpQuorumState

**Name attribute**

The Name persistent attribute is a user-defined name for the test resource. Both the aggregate and constituent resource will have the same value for this attribute.
A value for this attribute must be specified when a new IBM.Test resource is created and it must be unique.

**NodeNameList attribute**

The NodeNameList persistent attribute is an array of strings that indicates which nodes the IBM.Test resource is available on.
If the resource is floating, the TestRM will ensure that there is one constituent (fixed) resource for each node name in this list. Constituent resources are implicitly created or deleted as necessary to match the entries in this list. Constituent resources will only contain one entry in this array since they are fixed resources and thus only available on one node. This attribute for a floating resource is implicitly modified if a constituent resource is explicitly removed by the administrator so that the aggregate and constituent resource relationship is always consistent.
This list may be empty for a floating resource which means that it is not available anywhere and constituents may be added separately.

**ResourceType attribute**

You use the ResourceType persistent attribute to identify whether the resource is fixed or floating. An integer value of 0 indicates that the resource is fixed, a value of 1 indicates that it is a floating resource. This attribute defaults to fixed if not specified when a new IBM.Test resource is created.

**ForceOpState attribute**

You use this attribute to initiate an OpState change of the test resource via the RMC **chrsrc** command. This could be used to simulate a failure in the resource. The last state change is saved in this persistent resource attribute. Specifying this attribute during the creation of the resource has no effect. Normally ForceOpState changes should be done on constituent resources, as the aggregate resource collects the OpState of the whole resource. Allowed values of this attribute are:

Unknown=0

Online=1

Offline=2

Failed Offline=3

Stuck Online=4

Pending Online=5

Pending Offline=6

**TimeToStart** After a test resource receives the start command, the TimeToStart attribute specifies the amount of time (in seconds) it takes a resource to change its OpState from pending online to online. The default value is 0 seconds, then the resource immediately goes online.

**TimeToStop** After a test resource receives the stop command, the TimeToStop attribute specifies the amount of time (in seconds) it takes a resource to change its OpState from pending offline to offline. The default value is 0 seconds, then the resource immediately goes offline.

**WriteToSyslog** A resource of the class IBM.Test is capable to log online, offline and ForceOpState events in the Linux syslog facility. You use the WriteToSyslog attribute to turn on/off the writing to the syslog daemon. Allowed values of this attribute are:

**0** Do not write to syslog (this is default)

**1** Write to syslog

**OpState attribute**
The value of this dynamic state attribute contains the operational state of the resource. IBM.Test resource OpState follows the RMC start/stop commands or the ForceOpState event from an operator or test script (automated testcase). The possible values for this attribute as defined by the state transition diagram in the RMC architecture are:

Unknown=0

Online=1

Offline=2

Failed Offline=3

Stuck Online=4

Pending Online=5

Pending Offline=6

**MoveTime** Reserved for internal use.

**MoveFail** Reserved for internal use.

**MoveState** Reserved for internal use.

**OpQuorumState**
Reserved for internal use.

## Example: Create a test resource and manipulate its OpState

In order to create an IBM.Test resource on 2 nodes, issue the following RMC command:

```
mkrsrsc IBM.Test \
  Name="mytest" \
  NodeNameList="{'node01','node02'}" \
  ResourceType=1 \
  TimeToStart=5 \
  TimeToStop=2 \
  WriteToSyslog=1
```

The following command causes a constituent on node02 to change its OpState to Failed Offline. If the resource is automated by IBM Tivoli System Automation, the automation manager starts the resource on another node.

```
chrsrc -s "Name='myTest' && NodeNameList={'node02'}" IBM.Test ForceOpState=3
```

## Test resource manager

# Appendix A. Troubleshooting

Use this information to troubleshoot problems you might experience while using IBM Tivoli System Automation for Multiplatforms.

## Troubleshooting general base component errors and problems

This section covers the following topics:

- "How automation works"
  This topic summarizes important concepts of System Automation.
- "How to obtain troubleshooting information" on page 225
  Use this topic to learn how you can gather information about automated resources and resource groups.
- "Error analysis" on page 233
  Use the error scenarios described in this topic to learn how to understand, isolate, and resolve errors that are reported by System Automation. The following errors are discussed:
  - "A resource has an OpState of Failed Offline" on page 233
  - "A resource group has an OpState of Failed Offline" on page 234
  - "StartCommand timed out" on page 235
  - "StopCommand timed out" on page 236
  - "MonitorCommand timed out" on page 236
- "Problem analysis" on page 238
  Use the problem scenarios described in this topic to effectively troubleshoot problems related to System Automation that are typically not indicated by error messages. The following problems are discussed:
  - "A resource does not start" on page 238
  - "A resource group does not start" on page 239
  - "A resource does not stop" on page 239
  - "A resource group does not stop" on page 240
  - "No failover occurs after a node is excluded" on page 240
  - "No failover occurs after a node crash or reboot" on page 241

### How automation works

This topic summarizes important concepts of System Automation. For detailed information, refer to Chapter 7, "How IBM Tivoli System Automation processes the system information," on page 87.

#### Automation manager

The automation manager consists of the binder and the logic deck.

**Binder:** The binder is responsible for finding a placement for the members of a resource group when the group is started, or when the resources must be started on a different node because the node on which they were previously running has crashed or is rebooted. The corresponding task is the so-called binding step, and the result of the task is reflected in the BindingState of a resource.

The following BindingStates for a resource exist:

**Unbound**
The resource is not bound and therefore Offline. System Automation has not yet tried to find a placement for the resource.

**Bound**
The resource has been bound to a node, and the resource is either running on that node or System Automation will start the resource on the node after all dependencies on other resources have been fulfilled.

**Sacrificed**
System Automation could not find a placement for the resource. There is no node on which this resource could be started, which is why the resource will not be started by System Automation.

If the placement of more than one resource group results in a conflict, the priority value controls which group loses the conflict, which means that it is not placed and its binding state is set to Sacrificed. This is illustrated in the following example:

**Scenario:** In a two node cluster, resource group "RG1" contains resource " R1", and resource group "RG2" contains resource "R2". "R1" depends on "R2". Both are started. Subsequently, "R1" fails. The resulting behavior depends on the priority of the resource groups:

- If "RG1" and "RG2" have the same priority, "R1" is not restarted.
- If "RG1" has at least a priority of 21 (and "RG2" has a priority of 0), "R2" is stopped and "R2" and "R1" are started on a different node.

**Logic deck:** The logic deck is responsible for sending out the start and stop orders for the individual resources to bring them Online or Offline. When sending out the orders, the logic deck ensures that all start and stop dependencies defined in the automation policy are fulfilled.

## Important internal resource states

System Automation maintains information about many internal states for each resource. The most important are:

**DesiredState**
The state that System Automation anticipates for a resource is called DesiredState; it is the state the resource should be in when requests and votes from other resources are taken into account. The DesiredState is either Online or Offline.

The DesiredState of a resource is not necessarily identical to the value of the NominalState attribute of the resource group that contains the resource, because requests and votes that are submitted against a resource have a higher priority than the NominalState (for detailed information about request priorities, see "Using requests to start and stop resource groups and resources" on page 170).

**ObservedState**
The ObservedState is the actual state of the resource. It is monitored by the resource's resource manager, for example, with the MonitorCommand for a resource of class IBM.Application, and any state change is reported to the automation manager.

The goal of System Automation is to ensure that all resources' ObservedState values match their DesiredState.

## State cycle of a resource group

Figure 25 shows the state transitions that occur when System Automation successfully starts and stops a resource group.

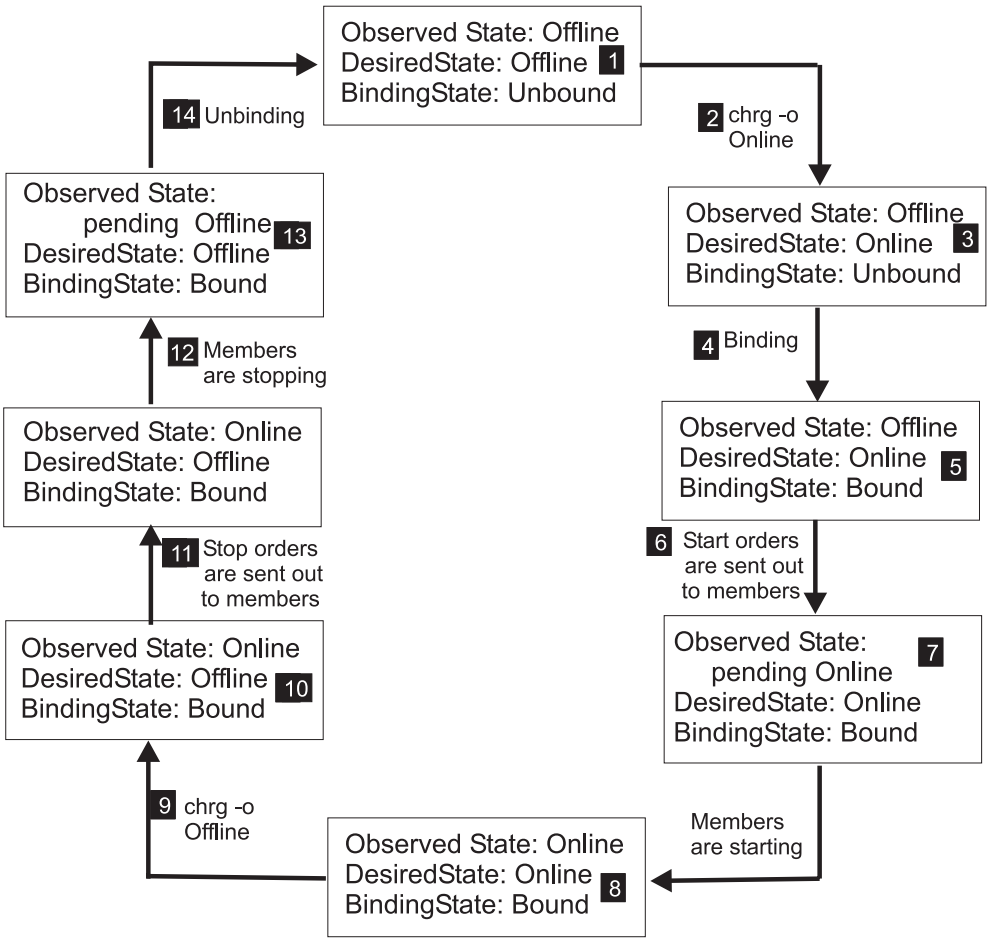



*Figure 25. State cycle of a resource group*

At the beginning of the cycle ( 1 ), the resource group's DesiredState is Offline, its ObservedState is Offline, and the members of the resource group are unbound, which results in a BindingState of Unbound.

If the NominalState of the resource group is changed to Online ( 2 ), the DesiredState of the resource group also changes to Online. The BindingState is still Unbound ( 3 ).

In the binding step ( 4 ), the binder tries to find a placement for all members of the resource group that satisfies all dependencies defined in the automation policy. If the binding step is successful, the BindingState of the resource group changes to Bound ( 5 ), otherwise the BindingState of the resource group will end up in state Sacrificed and the resources will not be started.

When all resource group members are bound, System Automation knows on which node to start them. The logic deck sends out start orders for the members of the resource group ( 6 ) and ensures that all dependencies defined in the automation policy are fulfilled. While the resources are starting, the ObservedState of the resource group changes to Pending Online ( 7 ).

When all resources are Online, the ObservedState of the resource group changes to Online ( 8 ).

When the NominalState of the resource group is changed to Offline ( 9 ), the DesiredState of the resource group changes to Offline ( 10 ). This will trigger System Automation to stop all members of the resource group. The logic deck sends out stop orders for the resources ( 11 ) and ensures that all dependencies defined in the automation policy are fulfilled. While the resources are stopping, the ObservedState changes to Pending Offline ( 12 ). When all resources are Offline, the ObservedState of the resource group changes to Offline ( 13 ).

In the unbinding step ( 14 ), the node placement of the resource group members is deleted from the internal tables and the BindingState of the resource group changes to Unbound ( 1 ).

# How to obtain troubleshooting information

## Before you begin: Collect debug and trace data

At any time of operation, the subsystems of System Automation and RSCT write debug and trace data to trace files on the local disk. The trace files are created as ring buffers to limit the amount of space taken up by the files. When the available space is exceeded, the trace files are overwritten. Depending on the number of resources and the activity on the nodes, large amounts of data may be logged to theses files and they may be overwritten at any time.

To ensure that no debug and trace data is lost and to increase the probability that all required diagnostic information is available should you need to contact IBM support, it is recommended that you collect all trace data before starting your troubleshooting activities, which may generate trace output.

To collect all debug and trace data, use the **ctsnap** command:

```
/usr/sbin/rsct/bin/ctsnap
```

The command generates a compressed file in the `/tmp` directory.

**Tip:** If you are sure that the problem is automation-related, for example, when a resource does not start, it is usually sufficient to collect the RecoveryRM trace. To save the RecoveryRM trace to a file, complete the following steps:

1. Find out on which node the RecoveryRM master daemon is running. To do this, issue the following command:

   ```
   lssrc –ls IBM.RecoveryRM | grep Master
   ```

2. Save the RecoveryRM trace to an output file. To do this, issue the following command on the master node:

   ```
   rpttr /var/ct/<domain-name>/log/mc/IBM.RecoveryRM/trace > <output-file-name>
   ```

## Using the system log as information source

The messages that are generated by all subsystems of System Automation and by RSCT are the first source of information in troubleshooting and problem determination:

- On **Linux**, messages are written to the system log (`/var/log/messages`).
- On **AIX**, the system logger is not configured by default. Messages are written to the error log.

  To be able to obtain the debug data, it is recommended that you configure the system logger in the file `/etc/syslog.conf`. When you have made the necessary changes, you must recycle the *syslogd* using the command **refresh –s syslogd**. The location of the log file is defined in `/etc/syslog.conf`.

Messages are logged to the system log for the following events:

- Start of a subsystem, for example:

  ```
  Mar 10 13:04:15 node1 RecoveryRM[5482]: (Recorded using libct_ffdc.a cv
  2):::Error ID: 824....zgK22/WNI/8cU0B.....................:::Reference
  ID:  :::Template ID: 0:::Details File:  :::Location:
   RSCT,IBM.RecoveryRMd.C,1.21.1.2,135           :::RECOVERYRM_INFO_0_ST
  IBM.RecoveryRM daemon has started.
  ```

- Stop of a subsystem, for example:

```
Mar 10 13:04:28 node1 RecoveryRM[5482]: (Recorded using libct_ffdc.a cv
2):::Error ID: 822....AhK22/osT18cU0B.....................:::Reference
ID:  :::Template ID: 0:::Details File:  :::Location:
RSCT,RecoveryRMDaemon.C,1.14,177              :::RECOVERYRM_2621_402_ER
IBM.RecoveryRM daemon stopped by SRC command or exiting due to an error
condition . Error id  0
```

- Error of a subsystem, for example:

```
Mar 10 13:04:14 node1 srcmstr: src_error=-9035, errno=0,
module='srchevn.c'@line:'251', 0513-035 The IBM.RecoveryRM Subsystem ended
abnormally. SRC will try and restart it.
```

- Messages related to the Quorum state of the subcluster, for example:

```
Mar  9 16:13:07 node1 ConfigRM[31411]: (Recorded using libct_ffdc.a cv
2):::Error ID: :::Reference ID:  :::Template ID: 0:::Details
File:  :::Location:
RSCT,PeerDomain.C,1.99.11.1,15510             :::CONFIGRM_HASQUORUM_ST The
operational quorum state of the active peer domain has changed to
HAS_QUORUM.  In this state, cluster resources may be recovered and
controlled as needed by  management applications.
```

- Start and stop of an IBM.ServiceIP resource, for example:

```
Mar  8 09:41:08 node1 GblResRM[1886]: (Recorded using libct_ffdc.a cv 2):::Error
ID: :::Reference ID:  :::Template ID: 0:::Details File:  :::Location:
RSCT,ServiceIP.C,1.2.5,1360                   :::GBLRESRM_IPONLINE IBM.ServiceIP
assigned address on device. IBM.ServiceIP 10.67.78.89 eth1:1

Mar  8 09:42:44 node1 GblResRM[1886]: (Recorded using libct_ffdc.a cv 2):::Error
ID: :::Reference ID:  :::Template ID: 0:::Details File:  :::Location:
RSCT,ServiceIP.C,1.2.5,1434                   :::GBLRESRM_IPOFFLINE
IBM.ServiceIP removed address. IBM.ServiceIP 10.67.78.89
```

- A StartCommand, StopCommand, or MonitorCommand for a resource of class IBM.Application runs into a timeout, for example:

```
Mar 13 10:25:55 node1 GblResRM[24275]: (Recorded using libct_ffdc.a cv
2):::Error ID: :::Reference ID:  :::Template ID: 0:::Details File:  :::Location:
RSCT,Application.C,1.2.1,2434                 :::GBLRESRM_MONITOR_TIMEOUT
IBM.Application monitor command timed out. Resource name resource1
```

**Tip:** In addition to the data that is logged by default, it is recommended that you log the execution of the StartCommand and StopCommand of IBM.Application resources to a specific location.

The scripts that are provided with the pre-canned policies for System Automation log all executions of the StartCommand and StopCommand for a resource to the system log by default. The following sample output shows the data that is written to the system log when a resource is started out of the StartCommand of the pre-canned policy for the NFS server:

```
Mar 13 10:34:31 node1 /usr/sbin/rsct/sapolicies/nfsserver/nfsserverctrl-
server:[27230]: NFS server started
```

## Using the audit trail as information source

The RecoveryRM daemon writes an audit trail entry for every order, for error responses to these orders, and for important information about the current policy. This trail can be viewed with the **rpttr** command:

```
rpttr /var/ct/<domain_name>/log/mc/IBM.RecoveryRM/trace_summary
```

**Example:**

The resource group A was started by an operator, which leads to starting its member RA. After a while, the resource group is stopped and an offline order is sent to its member:

```
12:16:20.168613 T(1096711088) _RCD Online request injected: A/ResGroup/IBM.ResourceGroup
12:16:20.181285 T(1096711088) _RCD Online Request against RA on node saxb02
12:16:35.722675 T(1096711088) _RCD Offline request injected: A/ResGroup/IBM.ResourceGroup
12:16:35.727970 T(1096711088) _RCD Offline Request against RA on node saxb02
```

## Using commands to gather information

Use the commands described in this section to obtain detailed information about resources and resource groups if the information provided in the system log was not sufficient to resolve the problem. For best results, you should invoke the commands in the sequence in which they are listed.
For detailed information about the commands, refer to the *IBM Tivoli System Automation for Multiplatforms Base Component Reference*.

The following commands can be used to gather information about resources and resource groups:

1. **lsrg –Ab –V –g <resource-group-name>**

   The command **lsrg –Ab** shows all information about the resource groups that are defined in the automation policy. Add the **-V** option to also display the most important automation details, including the DesiredState, ObservedState and BindingState. When analyzing the output, make sure to check if the value of the ConfigValidity attribute indicates a configuration problem (for more information about the ConfigValidity attribute, refer to "Dynamically verifying resources" on page 194).

   This example shows the information that is displayed for a resource group that is Offline:

```
node1:~ # lsrg -Ab -V -g rg1
Starting to list resource group information.

Displaying Resource Group information:
All Attributes
For Resource Group "rg1".

Resource Group 1:
        Name                              = rg1
        MemberLocation                    = Collocated
        Priority                          = 0
        AllowedNode                       = ALL
        NominalState                      = Offline
        ExcludedList                      = {}
        Subscription                      = {}
        Owner                             =
        Description                       =
        Instruction                       =
        ActivePeerDomain                  = domain1
        OpState                           = Offline
        TopGroup                          = rg1
        MoveStatus                        = [None]
        ConfigValidity                    =
        AutomationDetails[CompoundState] = Satisfactory
                          [DesiredState] = Offline
                        [ObservedState] = Offline
                         [BindingState] = Unbound
                      [AutomationState] = Internal
                       [StartableState] = Yes
                          [HealthState] = Not Applicable
```

   This example shows the information that is displayed for a resource group that is Online:

```
node1:~ # lsrg -Ab -V -g rg1
Starting to list resource group information.

Displaying Resource Group information:
```

```
All Attributes
For Resource Group "rg1".

Resource Group 1:
        Name                            = rg1
        MemberLocation                  = Collocated
        Priority                        = 0
        AllowedNode                     = ALL
        NominalState                    = Online
        ExcludedList                    = {}
        Subscription                    = {}
        Owner                           =
        Description                     =
        Instruction                     =
        ActivePeerDomain                = domain1
        OpState                         = Online
        TopGroup                        = rg1
        MoveStatus                      = [None]
        ConfigValidity                  =
        AutomationDetails[CompoundState] = Satisfactory
                         [DesiredState] = Online
                        [ObservedState] = Online
                         [BindingState] = Bound
                       [AutomationState] = Internal
                        [StartableState] = Yes
                           [HealthState] = Not Applicable
```

2. **lsrg –m**

   The command displays the operational state of all managed resources.

   **Example:**

```
node1:~ # lsrg –m

Displaying Member Resource information:
Class:Resource:Node[ManagedResource]  Mandatory  MemberOf  OpState
IBM.ServiceIP:ip1                     True       rg1       Online
IBM.Application:app1                  True       rg1       Online
IBM.Application:app2                  True       rg2       Offline
```

3. **lssamctrl**

   The command **lssamctrl** displays the global automation parameters, for example, the list of excluded nodes and the RetryCount, which specifies the maximum number of retries for the StartCommand when the resource does not start at the first attempt.

   **Example:**

```
node1:~ # lssamctrl
Displaying SAM Control information:

SAMControl:
        TimeOut               = 60
        RetryCount            = 3
        Automation            = Auto
        ExcludedNodes         = {}
        ResourceRestartTimeOut = 5
        ActiveVersion         = [2.1.0.1,Wed Mar  8 17:44:04 2006]
        EnablePublisher       = Disabled
```

4. **lsrgreq -L {-m}**

   The command **lsrgreq –L** lists all requests that were issued against resource groups. Such requests are either issued directly from the command line using the command **rgreq**, or implicitly from the operations console, or by an end-to-end automation manager. In the latter case, the `Source` of the request is 'Automation'.

   **Example:**

```
node1:~ # lsrgreq –L

Displaying Resource Group request information:
All request information
ResourceGroup Priority Action Source   NodeList  Active  UserID  MoveStatus
rg1           low      Start  Operator {}        Active          None
```

If the command is executed with the **-m** option, all requests against resource group members are displayed.

**Example:**

```
node1:~ # lsrgreq -L –m

Displaying Member Resource request information:
All request information

Member Resource 1:
        Class:Resource:Node[ManagedResource] = IBM.Application:app1
        Priority                             = low
        Action                               = Start
        Source                               = Operator
        ActiveStatus                         = Active
        UserID                               =
```

5. **lssam**

   The command **lssam** displays summary information about the operational states of the managed resources on a per node basis. It provides additional information about excluded nodes and requests that were issued against resources or resource groups.

   **Example:**

   In this example, the command **lssam** was used to find out why the resource 'app1' is offline although the nominal state of the resource group 'rg1', of which it is a member, is Online.

```
node1:~ # lssam
Online IBM.ResourceGroup:rg1 Nominal=Online
        |- Online IBM.ServiceIP:ip1
                |- Online IBM.ServiceIP:ip1:node1
                '- Offline IBM.ServiceIP:ip1:node2 Node=Excluded
        '- Offline IBM.Application:app1 Request=Offline
                |- Offline IBM.Application:app1:node1
                '- Offline IBM.Application:app1:node2 Node=Excluded
Offline IBM.ResourceGroup:rg2 Nominal=Offline
        '- Offline IBM.Application:app2
                '- Offline IBM.Application:app2:node2 Node=Excluded
```

   The output shows why 'app1' is offline although the nominal state of resource group 'rg1' is Online (*Nominal=Online)*:

   - Node 'node2' is in the list of excluded nodes (**Node=Excluded**), which is why all resources on the node, including 'app1', are offline.
   - An Offline request has been issued against 'app1' (***Request=Offline***), which explains why it is also offline on node 'node1'.

   Note that the most important information is highlighted in color in the output.

6. **lsequ –Ab**

   The command **lsequ –Ab** is used to display all resources of type "equivalency" that are defined in the automation policy. Typically, equivalencies are used to define the network interfaces that can be used by resources of type IBM.ServiceIP.

   When analyzing the output of the command, make sure to check if the value of the ConfigValidity attribute indicates a configuration problem (for more information about the ConfigValidity attribute, refer to “Dynamically verifying resources” on page 194).

**Example:**

```
node1:~ # lsequ –Ab
Displaying Equivalency information:
All Attributes

Equivalency 1:
        Name                                = eq1
        MemberClass                         = IBM.NetworkInterface
        Resource:Node[Membership]           = {eth0:node1,eth0:node2}
        SelectString                        = ""
        SelectFromPolicy                    = ANY
        MinimumNecessary                    = 1
        Subscription[Consumer,...]            = {[EEZ,All,None]}
        ActivePeerDomain                    = domain1
        Resource:Node[ValidSelectResources] = {eth0:node1,eth0:node2}
        Resource:Node[InvalidResources]     = {}
        ConfigValidity                      =
        AutomationDetails[CompoundState]    = Undefined
```

The attribute Resource:Node[ValidSelectResources] must contain resources, especially if a dynamic SelectString is used. Subsequently, the OpState of the valid resources must be checked:

```
# lsrsrc IBM.<MemberClass-attribute-value> Name NodeNameList OpState
```

7. **lsrel –Ab**

   The command **lsrel –Ab** is used to display all relationships that are defined in the automation policy.

   **Example:**

```
node1:~ # lsrel –Ab
Displaying Managed Relationship Information:
All Attributes

Managed Relationship 1:
        Class:Resource:Node[Source] = IBM.Application:app1
        Class:Resource:Node[Target] = {IBM.Application:app2}
        Relationship                = StartAfter
        Conditional                 = NoCondition
        Name                        = app1_StartAfter_app2
        ActivePeerDomain            = domain1
        ConfigValidity              =
```

   When analyzing the output, check the relationships for completeness (for example, both the `Source` and the `Target` of a relationship must be defined), and check if the value of the ConfigValidity attribute indicates a configuration problem (for more information about the ConfigValidity attribute, refer to “Dynamically verifying resources” on page 194).

8. **samdiag** (for deep analysis)

   The command **samdiag** is used to display detailed state information for an individual resource. The command can also be used to externalize all internal variables of a resource from the automation manager. The command is very helpful in problem analysis, but it is not intended for daily use because it generates a large amount of information.

   **Example:**

```
node1:~ # samdiag -g rg1

Displaying information for the following:
 Resource Group "rg1":

Diagnosis::Resource: rg1/ResGroup/IBM.ResourceGroup
    type: CHARM Resource Group
   Status -
       Observed: Online          - Available
```

```
       Desired: Online          - Requested Online
      (Nominal: Online          - Nominal State: Online)
    Automation: Idle            - CharmBase trigger linked
     Startable: Yes             - Resource is startable
       Binding: Bound           - Bound
      Compound: Satisfactory    - Satisfactory

Resource Based Quorum: Not Supported   - CharmBase trigger linked
  Members and Memberships:
      +---HasMember                      ---> app1/Fixed/IBM.Application/node1
      +---HasMember                      ---> ip1/Fixed/IBM.ServiceIP/node1
      +---bind/HasMember                 ---> app1/Float/IBM.Application
      +---bind/HasMember                 ---> ip1/Float/IBM.ServiceIP
 Group Constraint: Collocated
  Binding Constraints:
  Flags:
    None
  Orders:
    Outstanding Order: None            - Resource is Available
  Dependencies:
    Start: Satisfied
      +---InCluster                    ---> Cluster
    Stop: Satisfied
  Binding exceptions:
    None
  Static Relationships:
      +---InCluster                    ---> Cluster
  Dynamic Relationships:
      +---bind/HasMember               ---> app1/Float/IBM.Application
      +---bind/HasMember               ---> ip1/Float/IBM.ServiceIP"
```

To query the details for a specific resource, the command has to be executed as follows:

```
samdiag IBM.<resource-class-name>:<resource-name>:<node-name>
```

For example, for the resource 'app1' on node 'node1':

```
node1:~ # samdiag IBM.Application:app1:node1
Displaying information for the following:
 Resource "IBM.Application:app1:node1":

Diagnosis::Resource: app1/Fixed/IBM.Application/node1
    type: Fixed Resource
   Status -
       Reported: Online          - Online
       Observed: Online          - Online
        Desired: Online          - Requested Online
       (Nominal: Offline         - Defaulted: offline)
     Automation: Idle            - Idle - Online completed
      Startable: Yes             - Resource is startable
        Binding: Bound           - Bound
       Compound: Satisfactory    - Satisfactory

Resource Based Quorum: Not Supported   - CharmBase trigger linked
   Groups and Aggregates:
       <---HasMember                 ---- rg1/ResGroup/IBM.ResourceGroup
       <---bind/HasMember            ---- rg1/ResGroup/IBM.ResourceGroup
   Binding Constraints:
   Flags:
     None
   Orders:
     Outstanding Order: None       - Idle - Online completed
  Dependencies:
     Start: Satisfied
       +---RunsOn                  ---> node1/Node/IBM.PeerNode
     Stop: Satisfied
       <---HasMember               ---- rg1/ResGroup/IBM.ResourceGroup
   Static Relationships:
```

```
    +---RunsOn                    ---> node1/Node/IBM.PeerNode
Dynamic Relationships:
    <---bind/HasMember            ---- rg1/ResGroup/IBM.ResourceGroup"
```

# Error analysis

Use the error scenarios described in this topic to learn how to effectively troubleshoot errors that are reported by System Automation.

The following error scenarios are discussed in this section:



## A resource has an OpState of Failed Offline

This error has three possible causes:

**The cluster node is not Online**

If a cluster node is not Online, all resources that are defined on the node have an OpState of Failed Offline. In such a case, the problem is not resource- but node-related.

**The MonitorCommand of the resource returns with return code 3 (== Failed Offline)**

To find out if this is the case, you execute the MonitorCommand manually and check the return code of the command. Perform these steps:

1. Get the value of the MonitorCommand attribute for the resource:

   ```
   # lsrsrc –s 'Name="<resource_name>" ' IBM.Application Name MonitorCommand
   ```

2. Execute the MonitorCommand
3. Get the return code of the MonitorCommand:

   ```
   # echo $?
   ```

If the return code is 3 (Failed Offline), investigate why the MonitorCommand itself returns this value and resolve the problem. After the problem has been resolved, the resource should have an OpState of Offline.

**System Automation has set the resource to 'Failed Offline' because previous attempts to start the resource failed**

If the MonitorCommand returns 2 (Offline) but the resource has an OpState of 'Failed Offline', this indicates either that the execution of the StartCommand for this resource returned with an error (not 0 or timeout) or that System Automation could not start the resource within the number of attempts defined in the RetryCount attribute (see the description of the **lssamctrl** command above).

To investigate the problem, do this:

1. Check the system log for messages indicating a timeout for the StartCommand for this resource.
2. If there is no such message, check the appropriate log files for the application that is behind the resource. Identify and correct all problems.
3. Check the audit trail.

   The following audit trail entries indicate problems in the start script:

```
12:16:35.727970 T(1096711088) _RCD RMC
             Rejected online request against RA on node saxb02
12:16:35.727970 T(1096711088) _RCD
             Failed Offline Request against RA on node saxb02
```

The following entries indicate that the start command repeatedly ran into a timeout:

```
12:16:35.727970 T(1096711088) _RCD
              Maximum timer cancelled for RA on node saxb02
12:16:35.727970 T(1096711088) _RCD
              Failed Offline Request against RA on node saxb02
```

4. Finally, use the following command to reset the resource from the 'Failed Offline' state:

   ```
   # resetrsrc –s 'Name="<resource_name>" && NodeNameList={"node_name"}' \
   IBM.Application
   ```

   Now the resource should have an OpState of Offline and System Automation will start the resource again if the desired state of the resource is Online.

## A resource group has an OpState of Failed Offline

If the resources of a resource group do not start and the resource group has an OpState of 'Failed Offline' this indicates that the binder was unable to find a placement for the resources and furthermore the resource group's BindingState is Sacrificed. Check this using the following command:

```
# lsrg –Ab –V –g <resource_group_name>
```

If the BindingState is Sacrificed, do this:

- Check the audit trail for entries like in the following example:

  ```
  9:22:46.520729 T(229390) _RCD Online request injected:
              A/ResGroup/IBM.ResourceGroup
   09:22:46.522817 T(229390) _RCD RIBME-Hist for <NULL>:
              BINDER: Bind A/ResGroup/IBM.ResourceGroup
   09:22:46.532464 T(229390) _RCD RIBME-Hist for <NULL>:
              BINDER: Resource RA/Fixed/IBM.Test/saxb02 hsa no usable options
   09:22:46.532467 T(229390) _RCD RIBME-Hist for <NULL>:
              BINDER: Resource RB/Fixed/IBM.Test/saxb03 hsa no usable options
                      Resource RB/Fixed/IBM.Test/saxb02 hsa no usable options
                      Resource RA/Fixed/IBM.Test/saxb03 hsa no usable option
  ```

  The example shows a resource group that is collocated but has two fixed members on different nodes preventing the binder from placing the resources; this is expressed as 'having no usable option'.
- Check that there are no resources with OpState 'Failed Offline' within that group
- Check the ExcludedList attribute (lssamctrl) for excluded nodes
- Check that all relationships for starting the group are fulfilled
- Check that the equivalencies in the automation policy all have 'valid' and Online members:

  ```
  # lsequ –Ab
  ```

  – Check the attributes Resource:Node[ValidSelectResources] and MemberClass

    There should be 'valid' resources listed
  – Check the 'OpState' of the resources in the equivalency:

    ```
    # lsrsrc <resource_class> Name NodeNameList OpState
    ```

    where <resource_class> represents the MemberClass above

    An OpState of 1 indicates that the resource is Online, an OpState of 2 indicates that it is Offline.

## A resource has an OpState of Stuck Online

There are two possible reasons, why a resource has an OpState of Stuck Online:

- In rare cases, the MonitorCommand of a resource returns with return code 4 (== Stuck Online). This can be checked by executing the MonitorCommand manually and checking the command's return code:
  1. Retrieve the value of the MonitorCommand attribute for this resource:
     ```
     lsrsrc –s 'Name="<resource_name>" ' IBM.Application Name MonitorCommand
     ```
  2. Execute the MonitorCommand.
  3. Retrieve the return code of the MonitorCommand:
     ```
     echo $?
     ```

     If the return code is 4 (Stuck Online), investigate why the MonitorCommand itself returns this value. After this problem has been resolved, the resource should have an OpState of Offline.
- The second and more likely reason for a resource to have an OpState of Stuck Online (if the MonitorCommand returns 1 (Online) or 6 (Pending Offline), but the resource has an OpState of 'Stuck Online') is that a the resource could not be stopped by IBM Tivoli System Automation previously, and IBM Tivoli System Automation has finally set the resource to Stuck Online. This is the case if the execution of the StopCommand for this resource and a subsequent reset against that resource failed to bring the resource offline.

  This error cannot be recovered by IBM Tivoli System Automation and manual intervention is required. After investigating why the resource did not stop, an operator must stop the resource. When the OpState of the resource is evaluated as Offline at the next execution of the MonitorCommand, IBM Tivoli System Automation will again take control of this resource, and no further manual steps are required.

## Timeout messages are found in the system log

**StartCommand timed out:** A message is logged in the system log if the StartCommand for a resource did not finish within the time period defined in the StartCommandTimeout attribute for this resource. There are two possible causes for this problem:

- The value defined in the StartCommandTimeout attribute is to low

  To check the value of the attribute, perform these tasks:
  1. Determine the actual setting of this attribute for this resource:
     ```
     # lsrsrc –s 'Name="<resource_name>" ' IBM.Application Name \
     StartCommandTimeout
     ```
  2. Determine how long it takes to execute the StartCommand for this resource

     **Important:** This should never be done on a running production system, but only during maintenance, or on another test system!

     Keep in mind that the time needed to execute the command may increase if the system is under load.
  3. Compare the actual setting of the timeout value with the time needed to execute the command
  4. If necessary, adjust the StartCommandTimeout value:
     ```
     # chrsrc –c 'Name="<resource_name>" ' IBM.Application \
     StartCommandTimeout=<new_value_in_seconds>
     ```

     This change can be done dynamically.

- The StartCommand execution results in a hang situation, because one of the statements within the executed script hangs.
  - Investigation requires manual execution of the StartCommand

    **Important:** This should never be done on a running production system, but only during maintenance!
  - If it turns out that the script is not finishing (hanging), additional debugging can be activated by adding `set –x` as the second line of the StartCommand script.
  - Identify the statement that results in the hang and correct the problem. Note that this is out of the scope of System Automation.

**StopCommand timed out:** A message is logged in the system log if the StopCommand for a resource did not finish within the time period defined in the StopCommandTimeout attribute for this resource. There are two possible causes for this problem:

- The value defined within the StopCommandTimeout attribute is to low

  To check if this is the case, do this:

  1. Determine the actual setting of this attribute for this resource:

     ```
     # lsrsrc –s 'Name="<resource_name>" ' IBM.Application Name \
     StopCommandTimeout
     ```
  2. Determine how long it takes to execute the StopCommand for this resource

     **Important:** This should never be done on a running production system, but only during maintenance, or on another test system!

     Keep in mind that the time needed to execute the command may increase if the system is under load.
  3. Compare the actual setting of the timeout value with the time needed to execute the command.
  4. Adjust the SopCommandTimeout value if needed:

     ```
     # chrsrc –c 'Name="<resource_name>" ' IBM.Application \
     StopCommandTimeout=<new_value_in_seconds>
     ```

     This change can be done dynamically.
- The StopCommand execution results in a hang situation, because one of the statements within the executed script hangs.
  - Investigation requires manual execution of the StopCommand

    **Important:** This should never be done on a running production system, but only during maintenance!
  - If it turns out that the script is not finishing (hanging), then additional debugging can be activated by adding `set –x` as the second line of the StopCommand script.
  - Identify the statement that results in the hang and correct the problem. Note that this is out of the scope of System Automation.

**MonitorCommand timed out:** A message is logged in the system log if the MonitorCommand for a resource did not finish within the time period defined in the MonitorCommand attribute for this resource. There are two possible causes for this problem:

- The value defined within the MonitorCommandTimeout attribute is to low

  To check if this is the case, do this:

  1. Determine the actual settings of the MonitorCommand attributes for this resource:

```
# lsrsrc –s 'Name="<resource_name>" ' IBM.Application Name \
MonitorCommand MonitorCommandTimout MonitorCommandPeriod
```

2. Determine how long it takes to execute the MonitorCommand for this resource by issuing the MonitorCommand directly on the command line.

   Keep in mind that the time needed to execute the command may increase if the system is under load.

3. Compare the actual setting of the timeout value with the time needed to execute the command.
4. Adjust the MonitorCommandTimeout value if needed:

   ```
   # chrsrc –c 'Name="<resource_name>" ' IBM.Application \
    MonitorCommandTimeout=<new_value_in_seconds>
   ```

   This change can be done dynamically, however, the value of the MonitorCommandTimeout attribute must be lower than or equal to the value of the MonitorCommandPeriod attribute.

- The MonitorCommand execution results in a hang situation, because one of the statements within that script executed is hanging.
  - Investigation requires manual execution of the MonitorCommand
  - If it turns out that the script is not finishing (hanging), then additional debugging can be activated by adding `set –x` as the second line of the MonitorCommand script.
  - Determine the statement that results in the hang and correct the problem. Note that this is out of the scope of System Automation

# Problem analysis

Use this section to learn how to analyze and resolve the following problems:



## A resource does not start

If a resource does not start, perform the following steps:

1. Check for messages related to the execution of the StartCommand for that resource in the system log, the appropriate application log, and the process table (`ps –ef`). If the StartCommand has not been executed at all, proceed with Step 2, otherwise investigate why the application does not come Online.
2. Check for Operational Quorum:

   ```
   # lssrc –ls IBM.RecoveryRM | grep Quorum
   ```

   If Operational Quorum == HAS_QUORUM, proceed with Step 3. If not, find out how many nodes are Online, using:

   ```
   # lsrpnode
   ```

   Operational quorum requires that either more than half of the nodes in the cluster are online or exactly half of the nodes are online and the tie breaker has been reserved:

   - If less than half of the nodes are online, start additional nodes.
   - If exactly half of the nodes are online, check the attribute of the active tie breaker:

     ```
     # lsrsrc –c IBM.PeerNode OpQuorumTieBreaker
     ```

     If the value of this attribute is Operator, the tie breaker must set manually:

     a. Deny ownership of the tie breaker to the node that should not get it (if the other node is still Online):

        ```
        # runact –c IBM.PeerDomain ResolveOpQuorumTie Ownership=0
        ```

     b. Grant ownership of the tie breaker to the node that should get it:

        ```
        # runact –c IBM.PeerDomain ResolveOpQuorumTie Ownership=1
        ```

     The best way to ensure that the problem does not reoccur is to define an automatic disk or network tie breaker, which ensures that the tie breaker is reserved automatically.

     Now, check the setting for the active tie breaker:

     ```
     # lsrsrc –s 'Name="<name-of-active-tie-breaker>"' IBM.TieBreaker
     ```

     Check that the disk is correctly allocated for a disk tie breaker or that the IP address is available for a network tie breaker. Note that these settings can only be changed if **configuration quorum** is established, which means that more than half of the nodes must be online.
3. Check for requests against the resource or the containing resource group:

   ```
   # lsrgreq –L
   # lsrgreq –L –m
   # lssam
   ```

If a stop request was issued against the resource or resource group, this explains why the resource stays offline. To resolve the problem, cancel the request.

4. Check that the cluster is in automation mode and not in manual mode, which would prevent System Automation from starting resources, and that the node or nodes are not in the list of excluded nodes, because System Automation cannot start resources on excluded nodes. Use the following command:

   ```
   # lssamctrl
   ```

   If the value of the Automation attribute is Manual, the cluster is in manual mode. The mode can be set to Auto using:

   ```
   # samctrl –M F
   ```

   If there are nodes in the list of excluded nodes, they can be deleted from the list using:

   ```
   # samctrl –u d <node_name>
   ```

5. Check the DesiredState, ObservedState, and BindingState of the resource using this command for all nodes:

   ```
   # samdiag IBM.<resource-class>:<resource-name>[:<node-name>]
   ```

   If the BindingState for the resource is Sacrificed on all nodes, this indicates that the binder was not able to find a placement for this resource that satisfies all relationships to other resources. Typically, this problem occurs at the time when an automation policy is created or changed.

## A resource group does not start

A resource group is comprised of a number of resources. If none of the resources of the group is starting, perform the following steps:

1. Identify which of the resources should start first by evaluating the relationships.
2. Find out why this resource does not start by proceeding as described in section “A resource does not start” on page 238. Make sure to check for requests against the resource group, and evaluate all relationships in which the resource group is defined as source resource. To determine the BindingState of the resource group, use one of these commands:

   ```
   # lsrg –Ab –V –g <resource-group-name>
   # samdiag –g <resource-group-name>
   ```

## A resource does not stop

If a resource does not stop, perform the following steps:

1. If a resource does not stop after the StopCommand was executed, IBM Tivoli System Automation will issue a reset operation against the resource, which triggers the execution of the StopCommand. If the resource still does not reach the OpState Offline, the OpState of the resource will finally be set to Stuck Online. To resolve the problem, proceed as described in “A resource has an OpState of Stuck Online” on page 235.
2. If the StopCommand for the resource has not been executed, check for requests against the resource or the containing resource group, using one of these commands:

   ```
   # lsrgreq –L
   # lsrgreq –L –m
   # lssam
   ```

If there is a start request against the resource or resource group, check if the request can be canceled.

3. Check that the cluster is in automation mode, not in manual mode:

   ```
   # lssamctrl
   ```

   If the value for the attribute Automation is Manual, the cluster is in manual mode, and System Automation will not stop any resources. This can be set to Auto using:

   ```
   # samctrl –M F
   ```

4. Check if there are relationships from other resources that prevent this resource from stopping, especially, check for the following relationships:
   - StartAfter (a StartAfter relationship keeps the dependent resource online)
   - DependsOn and DependsOnAny (both relationships implicitly include a StartAfter relationship, which will keep the dependent resource online
5. Check if there are StopAfter relationships to other resources which prevent this resource from stopping (if the target resource needs to stay online, the source resource will also stay online).

## A resource group does not stop

A resource group is comprised of a number of resources. If none of the resources of the group is stopping, perform the following steps:

1. Identify which of the resources should stop first by evaluating the relationships.
2. Find out why this resource does not stop by proceeding as described in section “A resource does not stop” on page 239. Make sure to check for requests against the resource group, and evaluate all relationships in which the resource group is defined as target resource.

## No failover occurs after a node is excluded

If no failover occurs after a node is excluded, perform the following steps:

1. Check that the cluster is in automation mode and not in manual mode, and that a node is available on which the resources can be started using:

   ```
   # lssamctrl
   ```

   If the value for attribute Automation is Manual, the cluster is in manual mode, and System Automation will not start any resources. This can be set to Auto using:

   ```
   # samctrl –M F
   ```

   Display the list of nodes that are online and compare it with the list of excluded nodes using:

   ```
   # lsrpnode
   ```

   If there are too many or all nodes in the list of excluded nodes, you can delete nodes from the list using the following command:

   ```
   # samctrl –u d <node_name>
   ```

2. Check if there are resources with OpState ‘Failed Offline’. If there are, proceed as described in section “A resource does not start” on page 238.
3. Check if there are resources with OpState ‘Stuck Online’. If there are, proceed as described in “A resource has an OpState of Stuck Online” on page 235.
4. Check the BindingState of the resource group to be started first:
   - If the Binding State is 'Sacrificed', System Automation could not find a placement for the resources.

Check the equivalencies for valid member resources and check that the OpState of these resources is Online, using:

```
# lsequ –Ab
```

Check the attribute ValidSelectResources.

Check that the resources are Online:

```
# lsrsrc IBM.<equivalency_class> Name NodeNameList OpState
```

- If the BindingState is 'Bound', System Automation could not start the resources. Proceed as described in section "A resource does not start" on page 238.

5. Check for relationships that cannot be fulfilled without the excluded node.

## No failover occurs after a node crash or reboot

To analyze and resolve the problem, perform the following steps:

1. Check for operational quorum:

   ```
   # lssrc –ls IBM.RecoveryRM | grep Quorum
   ```

   If Operational Quorum == HAS_QUORUM, proceed with Step 2. If not, find out how many nodes are Online, using:

   ```
   # lsrpnode
   ```

   Operational quorum requires that either more than half of the nodes in the cluster are online or exactly half of the nodes are online and the tie breaker has been reserved:

   - If less than half of the nodes are online, start additional nodes.
   - If exactly half of the nodes are online, check the attribute of the active tie breaker:

     ```
     # lsrsrc –c IBM.PeerNode OpQuorumTieBreaker
     ```

     If the value of this attribute is Operator, the tie breaker must be set manually:

     a. Deny ownership of the tie breaker to the node that should not get it (if the other node is still online):

        ```
        # runact –c IBM.PeerDomain ResolveOpQuorumTie Ownership=0
        ```

     b. Grant ownership of the tie breaker to the node that should get it:

        ```
        # runact –c IBM.PeerDomain ResolveOpQuorumTie Ownership=1
        ```

     The best way to ensure that the problem does not reoccur is to define an automatic disk or network tie breaker, which ensures that the tie breaker is reserved automatically.

     Now, check the setting for the active tie breaker:

     ```
     # lsrsrc –s 'Name="<name-of-active-tie-breaker>"' IBM.TieBreaker
     ```

     Check that the disk is correctly allocated for a disk tie breaker or that the IP address is available for a network tie breaker. Note that these settings can only be changed if **configuration quorum** is established, which means that more than half of the nodes must be online.

2. Check that the cluster is in automation mode and not in manual mode, which would prevent System Automation from starting resources, and check that the node or nodes are not in the list of excluded nodes, because System Automation cannot start resources on excluded nodes. Use the following command:

   ```
   # lssamctrl
   ```

If the value of the Automation attribute is Manual, the cluster is in manual mode. The mode can be set to Auto using:

```
# samctrl –M F
```

If there are nodes in the list of excluded nodes, you can delete nodes from the list using the following command:

```
# samctrl –u d <node_name>
```

3. Depending on whether the entire resource group or only one of its members does not start, proceed as described in the appropriate section above.

### No reset occurs after a start control operation timeout

The operation timer is started when IBM Tivoli System Automation first sends a resource start control operation to a resource. If the resource does not reach the desired state (online) within the timeout period and IBM Tivoli System Automation fails to issue a reset operation against the resource, perform the following steps:

1. Stop the cluster using the **stoprpdomain** command.
2. Restart the cluster using the **startrpdomain** command.

### Other problems

If the problem is most likely related to the automation manager, you should try recycling the automation manager (IBM.RecoveryRM) before contacting IBM support. This can be done using the following commands:

1. Find out on which node the RecoveryRM master daemon is running using the following command:

   ```
   lssrc –ls IBM.RecoveryRM | grep Master
   ```

2. On the node running the master, retrieve the PID and kill the automation manager:

   ```
   lssrc –ls IBM.RecoveryRM | grep PID
   kill -9 <PID>
   ```

As a result, an automation manager on another node in the domain will take over the master role, and proceeds with making automation decisions. The *src* subsystem will restart the killed automation manager immediately.

## Reporting problems

Problems for which no troubleshooting information is available should be reported as PMRs against the product IBM Tivoli System Automation for Multiplatforms. When you report the problem, supply the following information:

- The debug and trace data you collected before troubleshooting (see “Before you begin: Collect debug and trace data” on page 225)
- A short description of the automation policy that was active when the problem occurred. To retrieve the required information you can use the command **sampolicy -s**.
- A short description of the tasks performed before the error occurred

# Troubleshooting the base component operations console

This section covers the following topics:



## An automation domain is not displayed in the topology tree

If a first-level automation domain is not visible in the topology tree in the operations console, perform the following steps:

1. Check if the adapter is running. To do this, issue the following command on one of the nodes of the domain:

   ```
   samadapter status
   ```

   If the adapter is running, a message like in the following example comes up:

   ```
   samadapter is running on sapb13
   ```

   If the adapter is automated, a message like in the following example comes up:

   ```
   Automated ResourceGroup 'samadapter-rg' runs on sapb13
   ```

   Make a note of the name of the node on which the adapter runs (in the example this is `sapb13`) and proceed with step 4.

   ---

2. If the adapter is not running, issue the following command to check if the domain is online:

   ```
   lsrpdomain
   ```

   A message like in the following example comes up:

   ```
   Name    OpState RSCTActiveVersion MixedVersions TSPort GSPort
   domain1 Online  2.4.4.2           No            12347  12348
   ```

   If `OpState` is not `Online`, start the domain.

   ---

3. If the domain is online, start the adapter with the following command:

   ```
   samadapter start
   ```

   After the start message has appeared, reissue the following command:

   ```
   samadapter status
   ```

   ---

4. If the adapter is running, check again in the operations console if the domain now appears in the topology tree.

   ---

5. If the domain is still not visible, you need the connection information that you specified in the adapter configuration dialog to resolve the problem.

   Perform the following steps:

   a. Launch the adapter configuration dialog of IBM Tivoli System Automation for Multiplatforms. To do this, issue the following command on a node in the domain:

      ```
      cfgsamadapter
      ```

      ---

   b. On the entry panel of the configuration dialog, click **Configure**.

c. Open the Adapter page on the Configure panel and record the values that appear in the following fields for future reference:
   - **Host name or IP Address**
   - **Request port number**

   This is the connection information the host running the operations console uses to reach the adapter on any of the nodes in the domain.

d. Open the page host using adapter and record the values that appear in the following fields for future reference:
   - **Host name or IP Address**
   - **Event port number**

   This is the connection information the adapter on any of the nodes in the domain uses to reach the host running the operations console.

6. Check if the operations console can be reached from each node in the domain. A simple test is `ping <operations console host>`.

   If there is a firewall between the nodes of the domain and the host running the operations console, check with the network administrator if the firewall permits a connection between the node (page Adapter: **Host name or IP Address**) and the host running the operations console (page Host using adapter: **Host name or IP Address** and **Event port number**).

7. The adapter determines whether SSL must be used for the communication with the operations console. To check the SSL settings of the adapter, launch the adapter configuration dialog using the command `cfgsamadapter`. On the Security page of the configuration dialog, verify that the SSL settings are correct. If the *Enable SSL* check box is selected, the operations console must also be configured to support SSL in direct access mode.

   Check this by running *./cfgdirect.sh* in the directory `<isc_runtime_root>/AppServer/profiles/default/Tivoli/EEZ`. Verify that the settings for the *eez-ssl-truststore*, *eez-ssl-keystore*, *eez-ssl-keystore-password*, and *eez-ssl-keystore-alias* keys used are the same as those as shown on the Security page of the configuration dialog. (For more information about the adapter configuration dialog, see the *IBM Tivoli System Automation for Multiplatforms Installation and Configuration Guide*.)

   You may not be able to stop the adapter using the **samadapter stop** command if the key chosen in adapter is incorrect or does not exist. In this case, use the following command to determine the process-ID:

   ```
   ps ax | grep sam.adapter
   ```

   Then use the following command to terminate the samdapter process:

   ```
   kill <process-ID>
   ```

8. On the host running the operations console, use **netstat** to find out if it is listening for events on the event port defined in **Event port number**.

   When the event port number is set to 2002 on a Windows host, **netstat** brings up a message like in the following example:

```
C:\>netstat
Active Connections
  Proto  Local Address    Foreign Address                      State
  ...
  TCP    E2EHOST:2002     sapb13.boeblingen.de.ibm.com:45688  ESTABLISHED
  ...
```

If **netstat** does not display any information about the event port defined in **Event port number**, open the file `/etc/hosts` (on Windows the file is located in `C:\WINDOWS\system32\drivers\etc\hosts`) and verify that the loopback address (`127.0.0.1`) is not related to the actual host name. The loopback address should be related to localhost only.
For example, the entry in `/etc/hosts` may look like the following:

```
127.0.0.1               localhost.localdomain localhost
```

---

9. Check if each node in the domain can be reached from the operations console. A simple test is `ping <hostname or IP Address>`.

   If there is a firewall between the host running the operations console and the nodes of the domain, check with the network administrator if the firewall permits a connection between the host running the operations console (page host using adapter: **Host name or IP Address** and **Request port number**) and the node (page Adapter: **Host name or IP Address**).

---

10. On the on node on which the adapter is running, use **netstat** to find out if it is listening on the event port defined in **Request port number**.

    For example, when the Request port number is set to 2001, **netstat** brings up a message like this on AIX and Linux hosts:

```
sapb13:~ # netstat -atn |grep 2001
tcp        0      0 9.152.20.113:2001       :::*                    LISTEN
```

---

11. When the communication between all ports has been established correctly (see the descriptions above), check whether the EEZ Publisher is running. The EEZ Publisher must be running on the master node of the Base component of IBM Tivoli System Automation for Multiplatforms

    To check if the Publisher is running, perform the following steps:

    a. Issue the following command on one of the nodes of the automation domain:

```
- issue lssamctrl
```

       If the Publisher is enabled, you will receive output like in the following example:

```
safli03:~ # lssamctrl | grep Publisher
EnablePublisher        = EEZ
```

    b. Issue the following command on the master node of the base component of IBM Tivoli System Automation for Multiplatforms (see "Administering the recovery resource manager" on page 15 how to find out the master node):

```
ps ax
```

       You should receive output like in the following example:

```
safli04:~ # ps ax | grep Publisher
25756 ?   S  0:00   TECPublisher
          /etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf EEZ
25757 ?   S  0:00   TECPublisher
          /etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf EEZ
25758 ?   S  0:00   TECPublisher
```

```
          /etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf EEZ
25759 ?   S  0:00   TECPublisher
          /etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf EEZ
```

c. Issue the following command on the IBM Tivoli System Automation for Multiplatforms node on which the adapter is running:

```
netstat
```

You should receive output like in the following example:

```
Safli03:~ # netstat -atn | grep 5539
tcp        0      0 :::5539                 :::*                    LISTEN
tcp        0      0 9.152.21.82:5539        9.152.20.92:32793       ESTABLISHED
```

If the Publisher is not running or communication on port 5539 cannot be established, perform the following steps:

a. Check that the file `/etc/Tivoli/tec/samPublisher.conf` contains the following entry:

```
#--SAMP-EEZ:
Publisher=EEZ
LibraryPath=libTECPublisher.so
ConfigPath=/etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf
```

b. Check that the file `/etc/opt/IBM/tsamp/sam/cfg/EEZPublisher.conf` contains the following entries:

```
ServerLocation=adapter_ip_address
ServerPort=5539
```

The value specified for `adapter_ip_address` in the file must match the value provided on the Adapter tab of the adapter configuration dialog.

---

12. If the domain still does not appear in the operations console, contact IBM support and provide diagnostic information:

a. On each node in the domain, find out where the trace files are located. The trace files can be found in the `/eez/logs` subdirectory of the Tivoli Common Directory. To find the path to the Tivoli Common Directory, issue the following command:

```
cat  /etc/ibm/tivoli/common/cfg/log.properties
```

The command returns the path to the Tivoli Common Directory, for example:

```
Tivoli_common_dir=/var/ibm/tivoli/common
```

This means that the trace files can be found in the following directory:

```
/var/ibm/tivoli/common/eez/logs
```

b. Use `tar` to package all files in the directory and provide the archive to IBM support.

---

## Time stamps displayed in the operations console are not in local time

The values that appear in the time stamps in the operations console are derived from the time zone settings of the operating system where the Integrated Solutions Console server is installed. If the time stamps are not in local time, check the time zone settings on your Integrated Solutions Console server.

Typically, you use the configuration tools that are provided with the operating system to change the time settings:

- On AIX, you can configure the time settings with the smit or smitty system configuration tool. Use the menu entries **System environments —> Change/Show Date and Time** to adjust the time settings.
- On SuSE Linux, you can use the yast2 or yast system configuration tools. Use the menu entries **System** -> **Date and Time** (SLES-9) or **System —> Set Time Zone** (SLES-8).
- On Red Hat Linux distributions, you can use the configuration tools redhat-config-time or system-config-time.
- On Windows, you can adjust the time settings with the Date and Time entry on the Control Panel.

You may have to restart your operating system for the changes to take effect.

**Tip:**

**AIX, Linux:** If you have modified the time zone settings as described above but the values displayed in the time stamps in the operations console are still inappropriate, you can set the environment variable TZ to resolve the problem.

**Examples:**

- To set the time zone for Berlin, Germany, use the following command:

  ```
  export TZ="Europe/Berlin"
  ```

- To set the time zone to US Eastern Standard Time, use the following command:

  ```
  export TZ="US/Eastern"
  ```

## After a node reboot, fixed resources show incorrect error states in the operations console

Fixed resources that are displayed in the operations console show incorrect error states after a node reboot. The correct states will be displayed once the resources are started on this node.

# Appendix B. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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
Mail Station P300
2455 South Road
Poughkeepsie New York 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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, 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

- IBM, AIX, DB2, NetView, Tivoli, Tivoli Enterprise, Tivoli Enterprise Console, WebSphere, and z/OS are trademarks of International Business Machines Corporation in the United States, other countries, or both.
- Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Other company, product, and service names may be trademarks or service marks of others.

# Index

## Special characters



## A



## B



## C



## D



## E



## F



## G

## S



## T



## U



## V



## W

# X

# Readers' Comments — We'd Like to Hear from You

**System Automation for Multiplatforms**
**Base Component**
**Administrator's and User's Guide**
**Version 2 Release 2**

**Publication No. SC33-8272-00**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: FAX (Other Countries): (+49)+7031-16-3456
- Send your comments via e-mail to: eservdoc@de.ibm.com

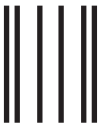
If you would like a response from IBM, please fill in the following information:

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | E-mail address |

**Readers' Comments — We'd Like to Hear from You**
SC33-8272-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape | **Please do not staple** | Fold and Tape



Cut or Fold
Along Line

Program Number: 5724-M00

Printed in USA

SC33-8272-00