High Availability Solution for WebSphere Message Queue in a single instance queue manager configuration using Tivoli SAfMP and a shared DASD device

Abstract

This paper describes the design and implementation of a high availability solution for a WebSphere® Message Queue Single Instance Queue Manager configuration on a pair of virtual Linux® servers running on an IBM System z10®. The solution tested during the creation of this paper used *Red Hat Enterprise Linux 5.5*, *Tivoli® System Automation for Multiplatforms 3.2*, and WebSphere MQ version 7.0.1. Although the solution described in this paper has been tested by the authors only with the software versions described, the same techniques are applicable to other Linux distributions and other versions of WebSphere MQ with some modifications.

This document assumes a basic working understanding of Linux on IBM System z® including IBM networking technologies, and WebSphere MQ operation and installation.

This document builds upon two earlier documents:

High Availability Solution for WebSphere Message Queue in a single instance queue manager configuration using Linux HA and a shared DASD device
http://www.**ibm.com**/systems/resources/sysserv_platformtest_mq_singleinstance.pdf

High Availability Solution for WebSphere Message Queue in a single instance queue manager configuration using HAE and a shared DASD device
http://www.**ibm.com**/systems/resources/HAS4WSMQ.PDF

Architecture at a glance

The WebSphere MQ high availability configuration consists of four resources, which will be managed by Tivoli System Automation for Multiplatforms:

1. WebSphere MQ Queue Manager – a WebSphere MQ system service that provides a logical container for the message queue, and is responsible for transferring data to other queue managers using message channels.
2. WebSphere Listener – a WebSphere MQ system service that accepts network requests from other queue managers, or client applications, and starts associated channels. Channels are used for distributed queuing.
3. A DASD device that contains the WMQ queue and log files, which is used as the persistent state shared between the two cluster nodes.
4. A virtual IP address, which serves as the public interface to the WMQ queue manager.

In this solution, two identically configured instances of WebSphere MQ are installed with one residing on each host system. The Queue and Log files used by the Queue Manager reside on a shared disk device that is made available to only one system at a time, which is referred to as the *active* system. The system that has control of the shared disk simultaneously has control of the virtual IP address used by the Queue Manager. In addition, this active system has an active copy of the Queue Manager running.

If the active system fails or is in the process of failing, the high availability manager stops the virtual IP address, stops the running copy of the Queue Manager, and unmounts the shared disk device if the system is intact enough for those actions to be taken. Control of all four resources, virtual IP address, shared DASD device, MQ listener, and running Queue Manager instance, is transferred to the *standby* system, at which time the shared disk is mounted, the IP address is made active, and the Queue Manager is started, thus enabling a new active system on the former standby system.

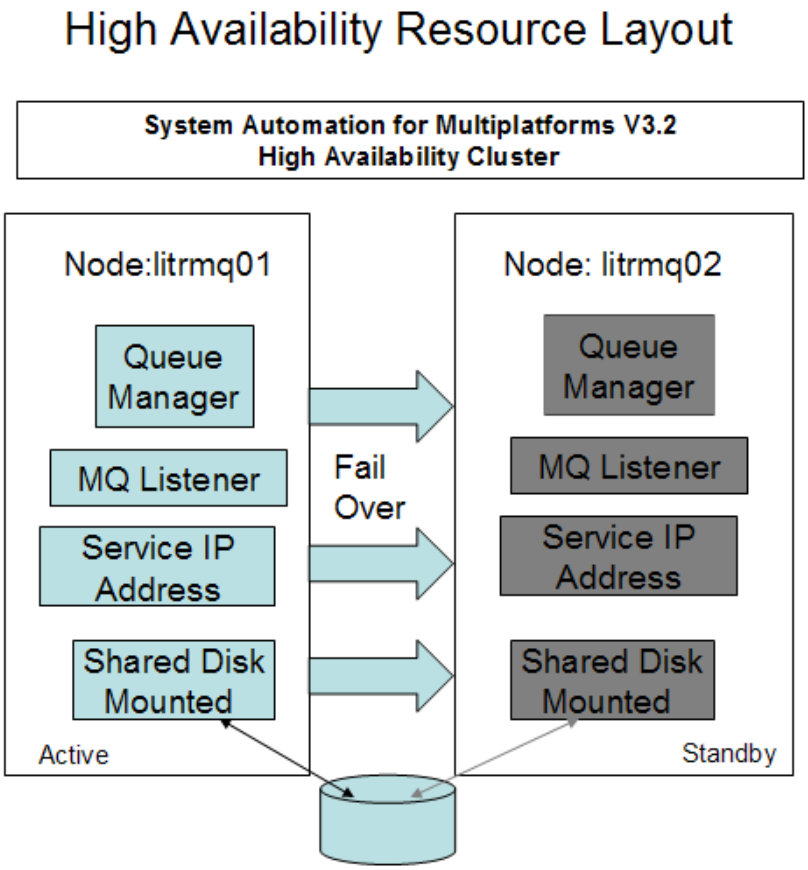

Figure 1: This diagram illustrates an example of our two node HA architecture. For this example, we have selected host names that match the documentation in the remainder of this whitepaper. In this diagram, node litrmq01 is in control of the High Availability Resources and is considered “active” while node litrmq02 is in standby mode and is the standby node. The direction of failover is shown to illustrate a single HA failover scenario. In reality, the failover is bi-directional.

Tivoli System Automation for Multiplatforms provides high availability and policy-based automation for applications and services across heterogeneous environments. The product provides the technologies required to ensures continuous high availability and reduce the frequency and duration of service disruptions for critical applications

Tivoli System Automation for Multiplatforms will be referred to as SAfMP in this document.

In addition to SAfMP, this solution also uses *System Automation for Multiplatform Automation Policies for LINUX*. This is a free offering made available on the IBM Tivoli Opal server. This offering is a collection of plug and play automation policy modules, developed to provide protection for critical business services and improve the availability of these business services. In this offering, automation policies for many IBM and non-IBM middleware and applications are provided.

Here is the link to the OPAL server:
http://www.**ibm.com**/software/brandcatalog/ismlibrary/details?catalog.label=1TW10SA02

SAfMP Automation Policies for Linux has recently been updated to support the current version of WebSphere MQ more completely. SAfMP Automation Policies for Linux will be referred to as AP in this document.

Some important SAfMP concepts

In the implementation steps, some SAfMP concepts will be mentioned that must be understood in order to understand the goal of the task that is being described. This section provides a brief description of those concepts.

Resource

A resource is any piece of hardware or software that can be defined to SAfMP. Resources can be defined using several methods. Two methods will be described in this document. One method is manually from the command line using the SAfMP **mkrsrc** command. SAfMP creates resources automatically for many system resources such as storage devices and network interfaces, using the built-in harvesting function. The harvesting or gathering function places the resources it discovers into appropriate resources classes.

Storage devices are categorized into the IBM.AgFileSystem resource class while network interfaces are placed into the IBM.NetworkInterface resource class.

See the *Tivoli System Automation for Multiplatforms Version 3.2 Administrator's and User's Guide* for more information on this topic.

Tiebreaker

A tiebreaker resource is a resource that is used to weight the vote of viable nodes when SAfMP is trying to determine if an Operation Quorum exits. The IBM.TieBreaker resource class can be configured to define one of a variety of tiebreaker resources. One tiebreaker resource type that is unique to Linux on System z is the use of an ECKD™ DASD device. In this solution a network tiebreaker resource was used. This will be discussed in detail later in this document.

See the *Tivoli System Automation for Multiplatforms Version 3.2 Administrator's and User's Guide* for more information on this topic.

Operational Quorum

SAfMP cannot safely perform automation if an *Operational Quorum* does not exist. Operational Quorum is based on the number of on-line nodes in a cluster. If there is an even number of nodes or the cluster has only two nodes, a tiebreaker resource is used to break the tie. Without an Operational Quorum, no automation takes place. In some situations within the cluster, such as a network problem that keeps some number of nodes from exchanging heartbeat signals, a condition called a *cluster split* occurs. If one

system can gain access to the tie breaker device, then that adds weight to the “vote” achieving Operational Quorum. SAfMP will take actions based on policy to make it safe to start resources on the surviving nodes. One such policy is the Protection Mode (CritRsrcProtMethod) policy, which is defined in the Configuration RM (IBM.ConfigRM) resource. A variety of options may be set as the Protection Mode option, including rebooting a resource that has control of a resource that must be moved in order to insure resource integrity.

See the *Tivoli System Automation for Multiplatforms Version 3.2 Administrator's and User's Guide* for more information on this topic.

Equivalency

An *equivalency* is a collection of resources that are capable of providing the same functionality. An equivalency could define, for example, a collection of network adapters that can be used to host an IP address used by an application resource that is capable of running on more than one node in the cluster. If a problem occurs requiring that the application be moved to another node, a network defined as an equivalency for that purpose will be selected to host the IP address.

See the *Tivoli System Automation for Multiplatforms Version 3.2 Administrator's and User's Guide* for more information on this topic.

SA operations console

The *system automation console* (SA operations console) is shipped with SAfMP and is an optional feature. It provides a graphical user interface that is quite useful in managing the cluster.
While this feature is optional, it is recommended that you install it. If installed, the SA operations console requires that you configure the end-to-end automation adapter.

See the *Tivoli System Automation for Multiplatforms Version 3.2 Administrator's and User's Guide* for more information on this topic.

End-to-end automation adapter

The *end-to-end automation adapter* is used by SAfMP to establish and manage communications between the automation domain and the SA operations console. The adapter is installed automatically but is not needed if you do not install the SA operations console, which is an optional feature but one we recommend.

See the *Tivoli System Automation for Multiplatforms Version 3.2 Administrator's and User's Guide* for more information on this topic.

Implementing the solution

Build two Linux systems

This solution is based on Red Hat Enterprise Linux 5.5.

Networking considerations

In this solution the high availability cluster contains two nodes, named litrmq01 and litrmq02.

You may want to consider a separate internal network device in addition to the default system network device, for each system to serve as the network interface for SAfMP communication. In the configuration described here, SAfMP used the primary system network device. We recommend using HiperSockets™ if deploying the HA solution within a physical host, or ideally using a VSWITCH or real OSA devices for the SAfMP network device when spanning hosts.

The systems will also each need a public IP address used to make WebSphere MQ available to the clients and applications requiring access to this service. This IP address will be implemented as a virtual IP address, and is active on only one of the two nodes at any given time.

Storage considerations

In addition to the storage required to contain the Linux system and WebSphere MQ, a single ECKD DASD device is required. If it is determined that more than one DASD device is required to contain your WebSphere MQ log data and log files, it would be easy to modify the configuration presented in this document to use an LVM or RAID configuration as the shared disk device. A z/FCP SCSI could also be used in place of the DASD device. Only the single DASD device configuration will be discussed here.

Shared disk considerations for z/VM

If both systems are on the same LPAR, that is they are guests of the same z/VM® system, then one Linux system will require a MDISK (mini disk) directory control statement in its z/VM system directory entry, while the other Linux system will require a LINK directory statement in its z/VM system directory entry. In our configuration we ran both Linux systems on the same z/VM system. Make sure that the MDISK is a full-pack minidisk when spreading a cluster across multiple z/VM systems. Only full-pack minidisks and dedicated devices will honor the required Virtual Reserve /Release commands across z/VM systems.

The system directory entry for litrmq01 would contain this MDISK directory control statement:

MDISK 0301 3390 6844 10016 DT0019 MWV

Parameter	Meaning
0301	The virtual address of the device
3390	The device type
6844	The starting cylinder for the minidisk.
10016	The number cylinders allocated to the minidisk.
DT0019	The volume serial number of the real DASD volume on which the minidisk resides.
MW	The mode of the minidisk. MW indicates Multiple-write access.
V	Tells CP to use its virtual reserve/release support in the I/O operations for the minidisk. MWV means the minidisk functions with write linkage using CP's virtual reserve/release.

The system directory entry for litrmq02 would contain this LINK directory control statement:

LINK LITRMQ01 0301 0301 MW

Parameter	Meaning
LITRMQ01	The user ID in the system directory, whose entry is to be searched for the device.
0301	The virtual address that was assigned to the device on litrmq01.
0301	The virtual device number that is to be assigned to the device on litsmq05.
MW	The mode of the device. MW indicates Multiple-write access.

In order to update the z/VM system directory, a user ID with at least Class B privileges is needed.

It is best to dedicate a one or more real DASD devices that will be shared by the individual Linux guests, when the guests reside on multiple LPARs. The shared disk device must be configured in the system configuration file for each z/VM system using an RDEVICE statement. This prevents z/VM from accidentally corrupting the contents of the device when it gets accessed from a different z/VM LPAR. This is applicable only when the cluster is spread across multiple z/VM systems.

Rdevice xxxx Type DASD Shared Yes MDC OFF

Parameter	Meaning
xxxx	This is real device address of the shared disk device.
Type DASD	The device type.
Shared Yes	Indicates that the device is to be shared between multiple real systems.
MDC OFF	Tells CP to not cache data on the real device in the minidisk cache.

With the disk online, it is then necessary to format the disk, partition it, and then make a file system before it can be used by Linux. For our testing we used the ext3 file system.

Install the required software

Install SAfMP on your system per the documentation provided with the installation media. Perform the basic configuration steps required to have a working two node SAfMP cluster.

SAfMP documentation can be found at this address:
http://publib.boulder.**ibm.com**/infocenter/tivihelp/v3r1/index.jsp?topic=%2Fcom.ibm.samp.doc_3.2.1%2Fwelcome.html

Install System Automation for Multiplatform - Automation Policies for Linux on your system.

The System Automation for Multiplatform - Automation Policies for Linux offering can be downloaded from the IBM Service Management Library.

Access to this site requires an IBM ID and password. You can learn about the IBM ID at this address:
https://www.**ibm.com**/account/profile/us?page=regfaqhelp

The binaries and documentation for System Automation for Multiplatform - Automation Policies for Linux can be found at this address:
http://www.**ibm.com**/software/brandcatalog/ismlibrary/details?catalog.label=1TW10SA02

Install WebSphere MQ on both systems. This solution is based on WebSphere MQ 7.0.1.0. Information on installing WebSphere MQ can be found at this address:
http://publib.boulder.**ibm.com**/infocenter/wmqv7/v7r0/topic/com.ibm.mq.amq1ac.doc/lq10220.htm

Configure a shared disk device

In this solution a shared disk device is used by the WebSphere MQ Queue Manager for log and queue data. This device will also contain the scripts that Pacemaker will use to start, stop, and monitor the resources. This solution uses an ECKD DASD device.

The shared DASD device must be configured to enable multiple systems to perform both read and write operations to the device. The integrity of the device will be maintained by SAfMP. SAfMP will insure that the device is mounted on only one system at a time.

Configure the shared disk for use by Linux

The steps described in this section can be performed on either of the nodes. They need to be performed only once, on a single node.

Determine the kernel device name for the shared DASD device. The Linux **lsdasd** command will provide this information.

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0200   active      dasda     94:0    FBA   512    1000MB    2048000
0.0.0201   active      dasdb     94:4    ECKD  4096   7042MB    1802880
0.0.0202   active      dasdc     94:8    ECKD  4096   7042MB    1802880
0.0.0301   active      dasdd     94:12   ECKD  4096   7042MB    1802880
0.0.0303   active      dasde     94:16   ECKD  4096   2347MB    601020
```

In the display shown above the kernel device name for device 0301, which is our shared disk device, is named dasdd.

Note: The kernel device name for the shared disk may not be the same on both nodes. This is not a concern. When this device is defined to SAfMP the by-path ID will be used. For DASD device 0301 the by-path ID is /dev/disk/by-path/ ccw-0.0.0301. After the 0301 device has been formatted and partitioned, the by-path ID will be /dev/disk/by-path/ ccw-0.0.0301-part1.

```
# ls /dev/disk/by-path
ccw-0.0.0200        ccw-0.0.0201-part1  ccw-0.0.0202-part1  ccw-0.0.0303
ccw-0.0.0200-part1  ccw-0.0.0201-part2  ccw-0.0.0301        ccw-0.0.0303-part1
ccw-0.0.0201        ccw-0.0.0202        ccw-0.0.0301-part1
```

Format the DASD device

The Linux **dasdfmt** command will format the shared disk device. In the example command the **–b** option is set to 4096. This is the blocking factor that will be used when the device is formatted. This is the IBM recommendation. Consult the storage administrator at your location to be sure that this meets location recommendations.

```
# dasdfmt -b 4096 -p -f /dev/dasdd
Drive Geometry: 10016 Cylinders * 15 Heads =  150240 Tracks

I am going to format the device /dev/dasdd in the following way:
   Device number of device : 0x301
   Labelling device        : yes
   Disk label              : VOL1
   Disk identifier         : 0X0301
   Extent start (trk no)   : 0
   Extent end (trk no)     : 150239
   Compatible Disk Layout  : yes
   Blocksize               : 4096

--->> ATTENTION! <<---
All data of that device will be lost.
Type "yes" to continue, no will leave the disk untouched: yes
Formatting the device. This may take a while (get yourself a coffee).

cyl 10016 of 10016 |#################################################| 100%

Finished formatting the device.
Rereading the partition table... ok
```

Partition the DASD device

Now that the shared disk device has been formatted, it must be partitioned. The Linux **fdasd** command is used to partition the device. The **–a** option tells the **fdasd** command to automatically create a single partition using all the space available on the device.

```
# fdasd /dev/dasdd -a
reading volume label ..: VOL1
reading vtoc .........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

Create the ext3 file system

The shared disk device has now been formatted and partitioned. The last step in preparing the device for use is to create a file system. In the testing of this solution, an ext3 file system was used. Because this is the only file system type that was tested, it is the one recommended for implementing this solution.

```
# mkfs.ext3 -b 4096 /dev/dasdd1
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
903168 inodes, 1802856 blocks
90142 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1849688064
56 block groups
32768 blocks per group, 32768 fragments per group
16128 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Set up the shared disk directory structure

Create a mount point for the shared disk.The /MQHA mount point must be created on both systems.

```
# mkdir -m 755 /MQHA
```

While the mount point must be created on both systems, the shared disk device should be mounted on only one system at a time.

Mount the shared disk device on one of the systems.

```
# mount /dev/dasdd1 /MQHA/
```

The Queue Manager requires a subdirectory for data (/MQHA/<queue_manager_name>/data. It also requires a directory for logs (/MQHA/<queue_manager_name>/log).

Create the subdirectories needed by the Queue Manager.

In this example the Queue Manager name is “QM1”.

```
# mkdir -m 755 /MQHA/QM1
# mkdir -m 755 /MQHA/QM1/data
# mkdir -m 755 /MQHA/QM1/log
```

Make the mqm user the owner of the subdirectories created in the prior step. Additionally, the mqm group will be made the owning group. The **–R** option on the Linux **chown** command makes the command recursive.

```
#  chown –R mqm:mqm /MQHA
```

You can verify the ownership of the subdirectories with the Linux **ls** command.

```
# ls -al /MQHA/QM1
total 16
drwxr-xr-x 4 mqm mqm 4096 Apr 13 14:06 .
drwxr-xr-x 3 mqm mqm 4096 Apr 13 14:05 ..
drwxr-xr-x 2 mqm mqm 4096 Apr 13 14:06 data
drwxr-xr-x 2 mqm mqm 4096 Apr 13 14:05 log
```

Create a Queue Manager for use in the HA cluster

Create a WebSphere Queue Manager instance on one of the nodes using the MQ **crtmqm** command. WebSphere MQ commands are usually required to be run under the MQ user ID. This is usually “mqm”. The Linux **su** command enables commands to be run with a substitute user ID. As used in the example below, the **–c** option on the **su** command indicates that a command string is to be executed. The command string is encapsulated in a pair of double quotes.

```
#su mqm -c "crtmqm -md /MQHA/QM1/data -ld /MQHA/QM1/log QM1"

WebSphere MQ queue manager created.
Directory '/MQHA/QM1/data/QM1' created.
Creating or replacing default objects for QM1.
Default objects statistics : 65 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

The MQ command **dspmqinf** used with the **command** option creates a command that can be used to populate the settings needed for the Queue Manager on the second node.
Run the **dspmqinf** command on the same node that the first instance of the Queue Manager was created on.

```
# su mqm -c "dspmqinf -o command QM1"
addmqinf -s QueueManager -v Name=QM1 -v Directory=QM1 -v Prefix=/var/mqm -v
 DataPath=/MQHA/QM1/data/QM1
```

The response from **dspmqinf** command is comprised of the MQ **addmqinf,** along with all of the information required to create a second copy of the original Queue Manager instance one the second node.

In preparation for applying the Queue Manager instance settings on the second node, copy the output of the **dspmqinf** command.

The shared disk device is no longer needed and should be unmounted using the Linux **umount** command:

```
# umount /MQHA
```

Mount the shared disk device at /MQHA on the second node using the by-path ID.

```
# mount /dev/disk/by-path/ccw-0.0.0301-part1 /MQHA/
```

Add Queue Manager configuration to the second node

Run the **addmqinf** command string created by the **dspmqinf** command on the active node. This command must be executed using the MQ user ID.

```
# su mqm -c "addmqinf -s QueueManager -v Name=QM1 -v Directory=QM1 –v
Prefix=/var/mqm -v DataPath=/MQHA/QM1/data/QM1"
WebSphere MQ configuration information added.
```

Configure SAfMP

We are configuring a two-node cluster. The environment variable “CT_MANAGEMENT_SCOPE” must be set to “2” on each node. This should be made part of the profile for each node. This can be done by placing a Linux **export** command in file /etc/profile on each node. You can do this using the **vi** editor or the editor of your preference.

This is the export command to be added to /etc/profile.

```
export CT_MANAGEMENT_SCOPE=2
```

Configure the automation cluster

The SAfMP **preprpnode** command sets up security for the cluster. A list of nodes is passed to the command. Security keys are exchanged between the nodes in the list. This enables communications between all nodes in the cluster. This must be done on every node in the cluster.

```
preprpnode litrmq01 litrmq02
```

Parameter	Meaning
litrmq01 litrmq02	List of nodes to that are members of the cluster

The SAfMP **mkrpdomain** command creates the cluster definition. This command can be run on any node and must be run only once.

```
#mkrpdomain mqha_domain litrmq01 litrmq02
```

Parameter	Meaning
mqha_domain	The name of the cluster
litrmq01 litrmq02	List of nodes to that are members of the cluster

Bring the new cluster on-line using the SAfMP **startrpdomain** command. This command can be run on any node and must be run only once.

```
#startrpdomain mqha_domain
```

Parameter	Meaning
mqha_domain	The name of the cluster

Verify the rpdomain and nodes are started using the SAfMP **lsrpdomain** command.

```
# lsrpdomain

Name       OpState RSCTActiveVersion MixedVersions TSPort GSPort
mqha_domain Online  2.5.5.1          No            12347  12348
```

Configure a tiebreaker resource

Configure a tiebreaker resource using the SAfMP **mkrscs** command.

```
# mkrsrc IBM.TieBreaker Type="EXEC" Name="mynetworktb"
DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=192.168.70.1 Log=1'
PostReserveWaitTime=30
```

Parameter	**Value Specified in Example**
IBM.TieBreaker	
Type	EXEC
Name	mynetworktb
DeviceInfo	PATHNAME=/usr/sbin/rsct/bin/samtp_net
Address	192.168.70.1
Log	1
PostReserveWaitTime	30

Set the active tiebreaker resource

The SAfMP **chrsrc** command is used to set the network tiebreaker as the active tiebreaker resource.

```
# chrsrc -c IBM.PeerNode OpQuorumTieBreaker="mynetworktb"
```

Parameter	**Value Specified in Example**	**Meaning**
-c	IBM.PeerNode	The resource class containing the attribute to be modified. In this case the attribute is OpQuorumTieBreaker.
OpQuorumTieBreaker	Mynetworktb	The tiebreaker resource to be made active.

Configure the cluster for the system automation console (optional)

Define the network interface equivalency for the end-to-end automation adapter

This is an optional task. If you do not install the SA operations console, this task does not need to be performed.

When SAfMP harvests network interfaces, it defines them as members of the IBM.NetworkInterface resource class. The harvest function further classifies the network interfaces into a resource group within IBM.NetworkInterfaces called the *CommGroup* group. All network interfaces configured with the kernel device name eth0 are placed in CommGroup "CG1" while all eth1 devices are placed in CommGroup "CG2", and so forth.

This task defines an equivalency resource called "netequ" which is used to define the equivalency for the network interface that SAfMP will use for communications within the cluster. The SAfMP **mkequ** command is used to create an equivalency resource.

```
# mkequ -D 'CommGroup=="CG1"' netequ IBM.NetworkInterface
```

Parameter	Value Specified in Example	Meaning
-D	'CommGroup=="CG1"'	The –D option specifies a dynamic string selection. In this case the command is searching for the CommGroup CG1.
	netequ	This is a positional parameter. It sets the unique name of the equivalency being created. In this example the equivalency is set to netequ.
	IBM.NetworkInterface	This is a positional parameter. It provides the name of the resource class containing the equivalency resource belongs to.

Configure the system automation adapter

The SAfMP command **cfgsamadapter** requires an x windows server or a VNC session to provide graphics support. The *System Automation for Multiplatforms Version 3.2 Installation and Configuration Guide* provides detailed instructions along with screen captures of the **cfgsamadapter** GUI. Please refer to section: Invoking the end-to-end automation adapter configuration dialog.

```
# cfgsamadapter
```

Start the system automation adapter

The SAfMP command **samadapter** can be run with the **start** option to start the adapter. This command can also be run with the **stop** option to stop the system automation adapter, or with the **status** option to determine the status of the system automation adapter.

```
# samadapter start
Automation started ResourceGroup 'samadapter-rg' waiting to become online...
ResourceGroup 'samadapter-rg' runs on litrmq01.ltic.pok.ibm.com
```

Configure System Automation for Multiplatform Automation Policies for LINUX

Edit the SAfMP Automation Policies configuration file

Copy /usr/sbin/rsct/sapolicies/mqm/sa-mq.conf.sample to /usr/sbin/rsct/sapolicies/mqm/sa-mq.conf.

```
#cp /usr/sbin/rsct/sapolicies/mqm/sa-mq.conf /usr/sbin/rsct/sapolicies/mqm/sa-mq
```

Edit /usr/sbin/rsct/sapolicies/mqm/sa-mq.conf with **vi** or your favorite editor. The file clearly delineates the customizable area.

```
###### START OF CUSTOMIZABLE AREA
################################################################################
#
# set default values
MQ_OWNER=mqm
MQ_MGR=QM1

# --directory for control scripts
script_dir="/usr/sbin/rsct/sapolicies/mqm"

# --prefix of all resources
prefix="SA-mq-"

# --list of nodes in the cluster
nodes="litrmq01 litrmq02"

# --IP address and netmask
#   If more instances of <ip_>, add more rows, like: ip_2 ip_3...
ip_1="192.168.71.175,255.255.255.0"

# --List of network interfaces ServiceIP ip_x depends on.
```

```
#   Entries are lists of the form <network-interface-name>:<node-name>,...
#   If more instances of <nieq_>, add more rows, like: nieq_2 nieq_3...
#   Default: create 'static equivalencies'
#   to create 'dynamic equivalencies' use keyword nieq_1_dyn ...
nieq_1="eth1:litrmq02.ltic.pok.ibm.com,eth1:litrmq01.ltic.pok.ibm.com"

# --common local mountpoint for shared data
#   If more instances of <data_>, add more rows, like: data_tmp, data_proj...
#   Note: the keywords need to be unique!
data_var="/MQHA"

# --LVM definitions: VG and optional hdisk  (only for AIX)
#   one entry allowed, like: myvg  ...  with hdisk like: myvg hdisk5
#lvm="VGNAME"
###### END OF CUSTOMIZABLE AREA
```

The frame above shows the customizable area of the sa-mq.conf file. The parameters that must be customized have been highlighted in bold.

The following table shows each parameter and the meaning of the value specified.

Parameter	**Value Specified in Example**	**Meaning**
MQ_OWNER	mqm	The MQ user ID.
MQ_MGR	QM1	The name of the queue manager.
prefix	SA-mq	A prefix to be used for all defined MQ resources. This is the default.
nodes	litrmq01 litrmq02	The members of the cluster.
ip_1	192.168.71.175, 255.255.255.0	The MQ virtual IP address and netmask.
nieq_1	eth1:litrmq01.ltic.pok.ibm.com, eth1:litrmq02.ltic.pok.ibm.com	The network interfaces that will be included in the MQ network interface equivalency group.
data_var	/MQHA	The mount point of the shared disk device.

Verify the System Automation for Multiplatform Automation Policies for LINUX configuration

Run the AP **cfgmq** command without any parameters, to display a list of resources that will be created when the command is run with the **–p** option, indicating that you want to perform the generated commands and build the resources.

```
generate command only : 'mkrg SA-mq-rg'
Generated definitions: 'SA-mq-mgr.def'
generate command only : 'mkrsrc -f SA-mq-mgr.def IBM.Application'
generate command only : 'addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-mgr'
Generated definitions: 'SA-mq-mq_lsn.def'
generate command only : 'mkrsrc -f SA-mq-mq_lsn.def IBM.Application'
generate command only : 'addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-mq_lsn'
Generated definitions: 'SA-mq-ip-1.def'
generate command only : 'mkrsrc -f SA-mq-ip-1.def IBM.ServiceIP'
generate command only : 'addrgmbr -m T -g SA-mq-rg IBM.ServiceIP:SA-mq-ip-1'
generate command only : 'mkrel -S IBM.Application:SA-mq-mq_lsn -G IBM.ServiceIP:SA-mq-ip-1 -p DependsOn SA-mq-mq_lsn-on-
ip-1'
generate command only : 'mkequ SA-mq-nieq-1 IBM.NetworkInterface:eth0:litrmq02.ltic.pok.ibm.com,eth0:litrmq01.ltic.pok.ibm.com'
generate command only : 'mkrel -S IBM.ServiceIP:SA-mq-ip-1 -G IBM.Equivalency:SA-mq-nieq-1 -p DependsOn SA-mq-ip-on-nieq-
1'
Generated definitions: 'SA-mq-data-var.def'
generate command only : 'mkrsrc -f SA-mq-data-var.def IBM.Application'
generate command only : 'addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-data-var'
generate command only : 'mkrel -S IBM.Application:SA-mq-mgr -G IBM.Application:SA-mq-data-var -p DependsOn SA-mq-mgr-on-
data-var'
generate command only : 'mkrel -S IBM.Application:SA-mq-mq_lsn -G IBM.Application:SA-mq-data-var -p DependsOn SA-mq-
mq_lsn-on-data-var'
generate command only : 'mkrel -S IBM.Application:SA-mq-mq_lsn -G IBM.Application:SA-mq-mgr -p StopAfter  SA-mq-mq_lsn-
StopAfter-mgr'

Generated resource definitions in: 'SA-mq-*.def'
and commands in script: 'SA-mq-make'.

Use script: 'SA-mq-make' to remove and create resources based on 'SA-mq-*.def' files.
```

The frame above shows the output from **cfgmq** command.

Create the System Automation for Multiplatform Automation Policies for LINUX resources based on the configuration file

Run the AP command **cfgmq** with the **–p** (perform) option.

```
# /usr/sbin/rsct/sapolicies/mqm/cfgmq -p

using option:  'perform commands!'.

Attention:
If resources matching 'SA-mq-*' exist, they will be removed.
Continue y|n?
y

successfully performed: 'rmrg SA-mq-rg'
successfully performed: 'mkrg SA-mq-rg'
successfully performed: 'rmrsrc -s 'Name=="SA-mq-mgr"' IBM.Application Force=1'
Generated definitions: 'SA-mq-mgr.def'
successfully performed: 'mkrsrc -f SA-mq-mgr.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-mgr'
successfully performed: 'rmrsrc -s 'Name=="SA-mq-mq_lsn"' IBM.Application Force=1'
Generated definitions: 'SA-mq-mq_lsn.def'
successfully performed: 'mkrsrc -f SA-mq-mq_lsn.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-mq_lsn'
Generated definitions: 'SA-mq-ip-1.def'
successfully performed: 'mkrsrc -f SA-mq-ip-1.def IBM.ServiceIP'
successfully performed: 'addrgmbr -m T -g SA-mq-rg IBM.ServiceIP:SA-mq-ip-1'
successfully performed: 'mkrel -S IBM.Application:SA-mq-mq_lsn -G IBM.ServiceIP:SA-mq-ip-1 -p DependsOn SA-mq-mq_lsn-on-
ip-1'
successfully performed: 'mkequ SA-mq-nieq-1 IBM.NetworkInterface:eth1:litrmq02.ltic.pok.ibm.com,eth1:litrmq01.ltic.pok.ibm.com'
successfully performed: 'mkrel -S IBM.ServiceIP:SA-mq-ip-1 -G IBM.Equivalency:SA-mq-nieq-1 -p DependsOn SA-mq-ip-on-nieq-1'
Generated definitions: 'SA-mq-data-var.def'
successfully performed: 'mkrsrc -f SA-mq-data-var.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-data-var'
successfully performed: 'mkrel -S IBM.Application:SA-mq-mgr -G IBM.Application:SA-mq-data-var -p DependsOn SA-mq-mgr-on-
data-var'
successfully performed: 'mkrel -S IBM.Application:SA-mq-mq_lsn -G IBM.Application:SA-mq-data-var -p DependsOn SA-mq-
mq_lsn-on-data-var'
successfully performed: 'mkrel -S IBM.Application:SA-mq-mq_lsn -G IBM.Application:SA-mq-mgr -p StopAfter  SA-mq-mq_lsn-
StopAfter-mgr'

Resource group, resources and relationships were removed and created.
```

The frame above shows the output from the **cfgmq –p** command. Reply **y** to enable the deletion of any existing resources.

Activate the MQ automation resources

Put the SA-mq-rg resource group on-line using the AP **chrg** command.

```
#chrg -o online SA-mq-rg
```

The SAfMP **lsrg** command can be used to verify the status of the SA-mq-rg resource group.

```
#  lsrg -g SA-mq-rg
Displaying Resource Group information:
For Resource Group "SA-mq-rg".


Resource Group 1:
        Name                                    = SA-mq-rg
        MemberLocation                          = Collocated
        Priority                                = 0
        AllowedNode                             = ALL
        NominalState                            = Online
        ExcludedList                            = {}
        Subscription[Consumer,EventFamily,EventFilter] = {[EEZ,All,None]}
        Owner                                   =
        Description                             =
        InfoLink                                =
        Requests                                = {}
        Force                                   = 0
        ActivePeerDomain                        = mqha_domain
        OpState                                 = Online
        TopGroup                                = SA-mq-rg
        ConfigValidity                          =
        TopGroupNominalState                    = Online
```

At this point, the SA-mq-rg resource group should be started and all resources should become active.

Management view of active configuration

The System Automation for Multiplatforms operations SA console was installed for use during testing. While it is an optional component, it is useful for managing the SAfMP cluster and seeing the status of the cluster and it's resources in a graphical format.

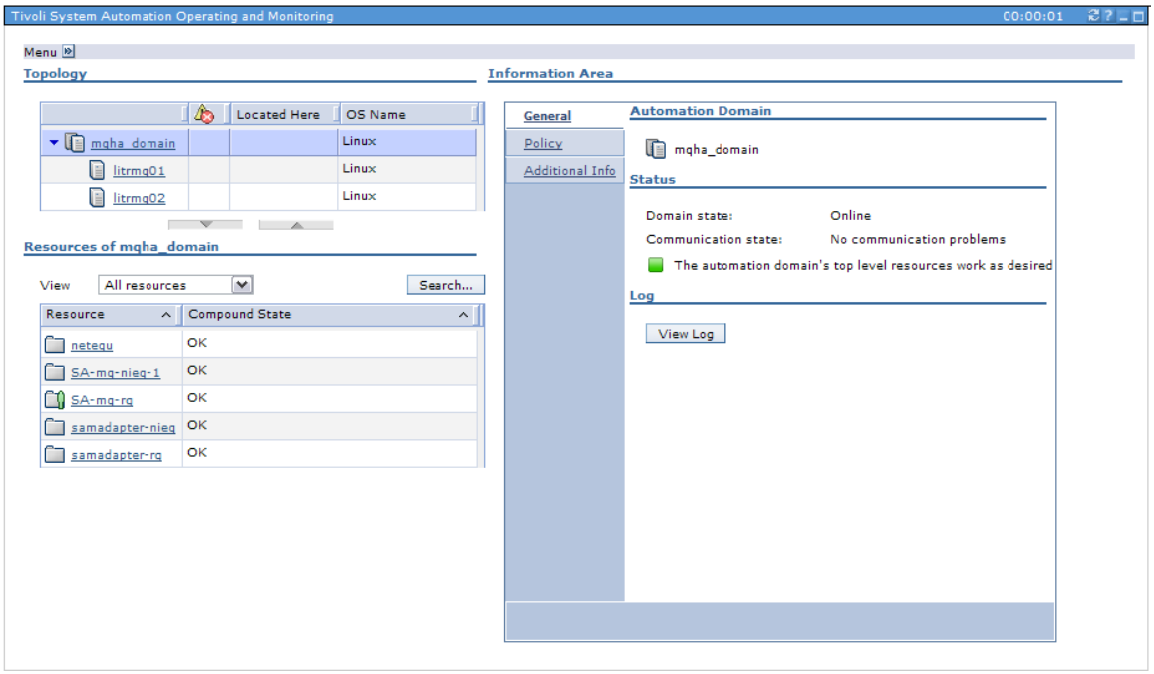

Figure 2: This is a management view of the test cluster, showing the status of the nodes and the resources that were managed by SAfMP.

Summary

The following use cases were tested:

- **Move the resource group** from the active system to the standby system.
  The resource group was moved by using the move function provided by the SA operations console. All resources were moved as expected.

- **Network failure.**
  Originally we tried to use a disk tiebreaker in our solution. That proved to be problematic when testing a network failure. The network failure was injected by executing the **service network stop** command on the z/VM console. We found that in the case of a network failure one node always was faster obtaining a reserve on the disk tiebreaker and therefore always kept control of the SA-mq-rg resource group.

  We decided to change our configuration to use a network tiebreaker and found that SAfMP was then able to execute a failover in the event of a network failure.

- **Storage device failure.**
  A storage device failure was injected by detaching the shared DASD device using the z/VM CP **detach** command from the z/VM console (#cp det 301). This caused the mqctrl-data script to unmount the shared device, so that the resources can be moved.

  Note: The default SAfMP /usr/sbin/rsct/sapolicies/mqm/mqcrtl-data script does not use the **lazy** unmount option. The section in the script that contains that form of the Linux **umount** command is commented out.

```
 ${UMOUNT} ${MountPoint} >/dev/null 2>&1
${UMOUNT} -f ${MountPoint} >/dev/null 2>&1
## lazy unmount taken out because of potential fs corruption
## can be included if needed, but introduces the risk again
##${UMOUNT} -l ${MountPoint} >/dev/null 2>&1
```

In my testing I found that using the **umount** command without the **lazy** (**-l**) option prevented the DASD device from being unmounted when the device was detached from the system. SAfMP was not able to perform a failover.

The SAfMP support team has found that at least one customer experienced a data integrity issue when using this option. While it is certainly safer to stay with the default script configuration, be aware that it is possible to change the script to use the **lazy** option on the **umount** for a more complete recovery in a similar failure scenario. However, this solution may introduce data integrity issues regarding the shared DASD device. Any decision to make a change here should be thoroughly tested before using this option in a production environment, and the decision should be made with the acknowledgement of the risks involved**.**

- **System failure.**
  This failure was injected by simply rebooting the active system from the z/VM console. SAfMP detected the failure and executed the failover process.

- **Queue manager failure.**
  This failure was injected by making file /var/mqm/sockets/@SYSTEM/qmgrlocl/litrmq01/Enqueue.lck unusable. This prevented the queue manager from starting until the file was deleted. MQ recreates this file if it is not found. SAfMP executed the failover process when it determined that the queue manager could not be started on litrmq01.

Results
In each case, the failover was detected and failover was complete in less than 5 minutes. This time was determined by comparing the time of the failure and injection and the time in the message log where the last member of the SA-mq-rg resource group was shown to be active.

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
07/2011