Unified Correlation Analyzer

Clustering and High-Availability Guide

Version 3.4
Edition: 1.0

Hewlett Packard
Enterprise

Notices

Legal notice

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for HPE products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

Printed in the US

Warranty

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Trademarks

Adobe®, Acrobat® and PostScript® are trademarks of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Java™ is a trademark of Oracle and/or its affiliates.

Microsoft®, Internet Explorer, Windows®, Windows Server®, and Windows NT® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Firefox® is a registered trademark of the Mozilla Foundation.

Google Chrome® is a trademark of Google Inc.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

X/Open® is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.

Red Hat® is a registered trademark of the Red Hat Company.

Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

Veritas™ Cluster Server is a registered trademark of Symantec Company.

Table of Contents

Figures

Tables

Preface

About this guide

This guide describes the clustering procedures for high-availability (HA) setup of the UCA for EBC product within a cluster.

Product Name: Unified Correlation Analyzer for Event Based Correlation (also referred to in this document as UCA for EBC)

Product Version: 3.4

Kit Version: V3.4

Intended Audience

Here are some recommendations based on possible reader profiles:

- Solution Developers
- Software Development Engineers
- System Integrators

Software Versions

The term UNIX is used as a generic reference to the operating system, unless otherwise specified.

The software versions referred to in this document are as follows:

Product Version	Supported Operating systems
UCA for Event Based Correlation Server Version 3.4	Red Hat Enterprise Linux Server release 6.5 & 7.2

Table 1 - Software versions

Typographical Conventions

`Courier` Font:

- Source code and examples of file contents.
- Commands that you enter on the screen.
- Pathnames
- Keyboard key names

Italic Text:

- Filenames, programs and parameters.
- The names of other documents referenced in this manual.

Bold Text:

- To introduce new terms and to emphasize important words.

Associated Documents

The following documents contain useful reference information:

[R1] *Unified Correlation Analyzer for Event Based Correlation Reference Guide*

[R2] *Unified Correlation Analyzer for Event Based Correlation Installation Guide*

[R3] Symantec Documents

[R4] Symantec VCS Home Page

[R5] RHEL 6 - Cluster Administration

Support

Please visit our HP Software Support Online Web site at https://softwaresupport.hpe.com/ for contact information, and details about HP Enterprise Software products, services, and support.

The Software support area of the Software Web site includes the following:

- Downloadable documentation.
- Troubleshooting information.
- Patches and updates.
- Problem reporting.
- Training information.
- Support program information.

Chapter 1
Introduction

This guide describes the recommended way to make UCA for EBC application Highly Available.

Basically, this recommended way is to set up a cluster (the term cluster refers to multiple independent systems connected into a management framework); to install UCA for EBC application on several nodes of this cluster, and to rely on the tools provided by the cluster manager to ensure the high availability of the UCA-EBC application.

Currently, the **Veritas Cluster Server** (VCS) from Symantec Company; and the **Red Hat Enterprise Linux Cluster** are the clusters on which the high-availability solutions described in this document are based on. It does not mean that UCA for EBC product is supported only on VCS, or RHEL cluster, but solely that other solutions are not yet fully described.

This guide describes the clustering procedures for high-availability (HA) setup of the UCA for EBC product within a cluster.

This guide provides instructions on how to configure UCA for EBC server to be managed for HA within a cluster as an HA resource. UCA for EBC server will be managed through the specific cluster manager.

However, please refer to the *Unified Correlation Analyzer for Event Based Correlation Installation Guide* [R2] for a better understanding on how to install UCA for EBC server in order to support HA mode.

1.1 Introducing UCA for EBC server

The UCA for EBC product offers generalized event based correlation solution. It is based on JBoss Drools engine. As such, the server delivered with the product is a Java process running on the supported platform.

Since V2.0 of the product, you can have multiple Java processes named "instances" running on a single host, which means one process per instance.

This version V3.4 brings the scripts to handle a proper HA solution based on VCS.

For more information on the UCA for EBC product, please refer to the *Unified Correlation Analyzer for Event Based Correlation Reference Guide* [R1].

1.2 Making UCA for EBC highly available

The recommended way to make UCA for EBC application Highly Available (HA) is to rely on a **High Availability (HA) cluster software**.

The examples of HA cluster software described in this guide are
- the Veritas Cluster Server (VCS)
- the RHEL Cluster High Availability Add-On

The idea is to install UCA for EBC on two or more nodes of a cluster, and to ask the HA cluster software to check the liveliness of the active UCA for EBC application(s).

If one UCA for EBC application is down, then the HA cluster software immediately brings up another instance of the UCA for EBC application.

The HA cluster software, through means of Virtual IP address, makes sure that other applications interacting with the UCA for EBC, such as Open Mediation or the Unified Mediation Bus (UMB) framework components, always see the same IP address for the active UCA for EBC application.

The HA cluster software also makes sure that the disk data (configuration, logs, Value Packs, ...) used by the UCA for EBC application which fails, is made available through mounting, to the UCA for EBC application that takes over.

The figure below illustrates a simple HA cluster configuration. It has only two nodes. On one node runs the active UCA for EBC application, on the other node, the UCA for EBC application is ready to be awaken if needed.

The Open Mediation application is configured to interact with the Virtual IP@ of the active UCA for EBC application.

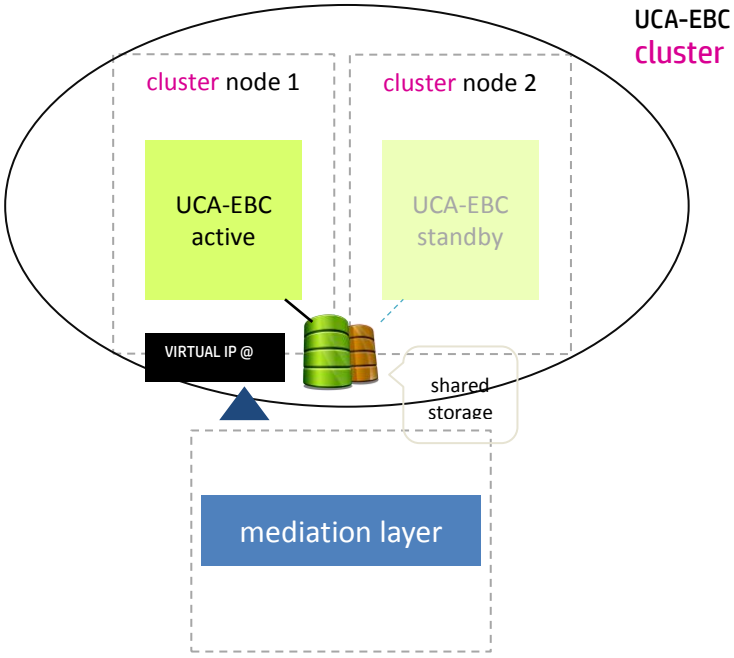

Figure 1 - The HA UCA-EBC service on a two-node cluster

In this guide, the examples taken are based on this simple two-nodes configuration.

In this simple configuration, there is one active UCA for EBC application, and one standby.

The description of more complex configurations with m active nodes and n standby nodes, is not addressed in this guide.

The ways to make the shared storage redundant are out of the scope of this guide.

Chapter 2
Installation of UCA for EBC

On Linux, the UCA for EBC Server product is delivered as a tar file named:

uca-ebc-server-kit-3.4-linux.tar

For detailed installations instructions and license setup, refer to the *Unified Correlation Analyzer for Event Based Correlation Installation Guide* [R2].

Briefly:

- Make sure that user uca is well defined (same id on all members of your cluster)
- Untar the tar file into a temporary directory
- Launch the installation script making sure you have specified a mounted NFS directory for the –d option, allowing to have the data directory used by UCA for EBC server to be accessible from all the nodes of your cluster, e.g.:

```
# ./install-uca-ebc.sh –d /var/shared/UCA-EBC
```

- Update the ~uca/.bashrc file adding following lines (per example):

```
# cat ~uca/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
[ -f /opt/UCA-EBC/.environment.sh ] && . /opt/UCA-EBC/.environment.sh

JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk.x86_64
export JAVA_HOME
```

The environment variables chosen at installation time are defined in $UCA_EBC_HOME/.environment.sh and this file has to be sourced by uca user in order to have the various scripts to work correctly.

Also, UCA for EBC server needs as a minimum Java 1.8 JRE (Java Runtime Environment) and have JAVA_HOME defined accordingly.

Chapter 3

The UCA for EBC HA solution based on VCS

3.1 Introducing Veritas Cluster Server (VCS)

Veritas™ Cluster Server (VCS) by Symantec provides High Availability (HA) and Disaster Recovery (DR) for mission critical applications running in physical and virtual environments. VCS ensures continuous application availability despite application, infrastructure or site failures. When a node or a monitored application fails, other nodes can take predefined actions to take over and bring up services elsewhere in the cluster.

An application service is a collection of hardware and software components required to provide a service where an end-user or application may access by connecting to a particular network IP address or host name.

Each application service typically requires components of the following three types:

- Application binaries
- Network
- Storage

VCS uses agents to monitor an application and brings bundled agents to manage a cluster's key resources.

Resources are VCS objects that correspond to hardware or software components, such as disk groups, logical volumes, and network interface cards (NIC), IP addresses, and applications.

The implementation and configuration of bundled agents vary by platform.

For more information about bundled agents, refer to the *Veritas Cluster Server Bundled Agents Reference Guide*. [R3]

The present document is based upon VCS 6.0.2 product. Refer to VCS release notes [R3] for more information on this particular product.
Note that most of commands given within this document should be applicable to earlier versions of VCS (e.g. 5.x).

3.1.1 System requirements

VCS is supported on Red Hat Enterprise Linux 6 Update 1, 2, 3 and on a 64-bits only chipset. If your system is running an older version of Red Hat Enterprise Linux, upgrade it before attempting to install the Veritas software.

VCS is not supported on HP-UX.

VCS requires that all nodes in the cluster use the same processor architecture and run the same operating system version. However, the nodes can have different update levels for a specific RHEL version.

3.1.2 How VCS detects failures and ensures HA

VCS detects failure of an application by issuing specific commands, tests, or scripts to monitor the overall health of an application. VCS also determines the health of underlying resources by supporting the applications such as file systems and network interfaces.

The scripts to monitor UCA for EBC server are delivered along with UCA for EBC product.

When VCS detects an application or node failure, VCS brings application services up on a different node in a cluster.

For more information about failures detection, refer to *Veritas Cluster Server Administrators Guide*. [R3]

3.2 Installing VCS

You will need to download the VCS from the Symantec VCS home page [R4]. Typically, you'll get a tar file named:

VRTS_SF_HA_Solutions_6.0.2_RHEL.tar

VCS can be downloaded as trial ware and comes with 60 days of free usage. Refer to Symantec VCS home page for getting a proper license.

For detailed installation instruction, refer to the *Veritas Cluster Server Installation Guide*. [R3]

As an example, this guide will use systems carlton0 and carlton1 as hostnames of the 2 members of the VCS cluster.

Briefly:

- Untar the rhel6 distribution from the VCS tar file into a temporary directory.
- Make sure all members of your cluster have the same clock and have correctly set the NTP configuration.
- Run the installer and verify that you cluster is conform to VCS needs, e.g.:

```
# ./dvd1-redhatlinux/rhel6_x86_64/cluster_server/installvcs -precheck
carlton0 carlton1
```

- In case of problems, check the logs in /opt/VRTS/install/logs

3.3 Configuring VCS

This chapter describes the configuration to be performed to have UCA for EBC server well configured within the VCS HA solution.

3.3.1 Setting up a VCS cluster

The VCS installation script allows you to directly configure your cluster once installation is done.

If not done during installation, run the configuration tool afterwards:

```
# /opt/VRTS/install/installvcs602 –configure
```

As an example, UCA for EBC has been validated with following configuration:

- I/O fencing disabled
- Cluster name = uca-cluster
- Heartbeat links using LLT over Ethernet
    1. Private Heartbeat NIC = eth1 (1000Mb/s)
    2. Low-Priority Heartbeat NIC = eth0 (100Mb/s)
- Cluster ID = 5020
- Virtual IP not configured

- Secure mode disabled
- SMTP and SNMP notifications disabled

After successful configuration of VCS, the VCS ha processes are running with an empty configuration :

```
# cat /etc/VRTSvcs/conf/config/main.cf
include "OracleASMTypes.cf"
include "types.cf"
include "Db2udbTypes.cf"
include "OracleTypes.cf"
include "SybaseTypes.cf"

cluster uca-cluster (
        UserNames = { admin = aPQoPMpMPjPNoWPrPX }
        Administrators = { admin }
        )

system carlton0 (
        )

system carlton1 (
        )
```

For subsequent command lines throughout this document, make sure the root $PATH contains the following path:

```
# export PATH=$PATH:/opt/VRTSvcs/bin
```

3.3.2 Creating a service group

A service group is a virtual container that enables VCS to manage an application as a unit. It contains all the hardware and software components required to run the service. The service group enables VCS to coordinate failover of the application service resources in the event of failure or at administrator's request.

A service group is a logical grouping of resources and resource dependencies. It is a management unit that controls resource sets. It is made up of resources and their links which you normally requires to maintain the HA of application.

Here we are going to configure an HA group to handle the UCA for EBC application.

Run the following commands to:

- Switch the configuration in read-write mode
- Create an HA group named "uca-group"
- Populate SystemList attribute so that the group is configured for all hosts of the cluster
- Enable automatically group on a preferred host
- Validate, apply the configuration and switch it to read-only mode

```
# haconf –makerw

# hagrp –add uca-group

# hagrp -modify uca-group SystemList carlton0 0 carlton1 1

# hagrp -autoenable uca-group –sys carlton0
```

```
# haconf -dump -makero
```

3.3.3 Creating HA resources for UCA for EBC server

There are multiple types of HA resources handled by VCS software. Here, we are going to focus on an HA resource that is going to be handled by the VCS **Application Agent**. The VCS Application agent is the only agent capable to monitor UCA for EBC server process, because it can make use of UCA for EBC specific scripts delivered with V3.4of UCA-EBC for this VCS usage.

Another mandatory HA resource will be the virtual IP address that is going to be used externally to access any member of the cluster transparently. This resource is handled by the VCS **IP Agent**.

Create the application resource

An application resource needs mandatory attributes to specify what are the scripts to use to start/stop/monitor a specific program. Here we are using the scripts delivered by UCA for EBC product.

Here below we are supposing that the configuration is in read-write mode.

Run the following commands to:

- Switch the configuration in read-write mode
- Create an HA resource named "uca-ebc" to handle UCA for EBC server process
- Configure the HA resource to use UCA for EBC provided scripts and make sure UCA for EBC will be launched by uca user.
- Enable the HA resource
- Validate, apply the configuration and switch it to read-only mode

```
# haconf -makerw

# hares -add uca-ebc Application uca-group

# hares -modify uca-ebc User uca
# hares -modify uca-ebc StartProgram "/opt/UCA-EBC/bin/uca-ebc-vcs
start"
# hares -modify uca-ebc StopProgram "/opt/UCA-EBC/bin/uca-ebc-vcs
stop"
# hares -modify uca-ebc MonitorProgram "/opt/UCA-EBC/bin/uca-ebc-vcs
monitor"
# hares -modify uca-ebc CleanProgram "/opt/UCA-EBC/bin/uca-ebc-vcs
clean"

# hares -modify uca-ebc Enabled 1

# haconf -dump -makero
```

After successful configuration of VCS HA group and resource, the VCS configuration has been updated with:

```
# tail -22 /etc/VRTSvcs/conf/config/main.cf
group uca-group (
        SystemList = { carlton0 = 0, carlton1 = 1 }
        )
```

```
        Application uca-ebc (
         User = uca
         StartProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs start"
         StopProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs stop"
         CleanProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs clean"
         MonitorProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs monitor"
         )



        // resource dependency tree
        //
        //   group uca-group
        //   {
        //   Application uca-ebc
        //   }
```

Above configuration is performed to launch the default instance of the UCA for EBC server program. If you want VCS to handle multiple instances, you should add as argument to all programs the name of the instance to launch. You can create as much resource as UCA for EBC instances that you want to monitor in your cluster.

Here below an example given for an instance called "bis".

```
# tail -31 /etc/VRTSvcs/conf/config/main.cf
group uca-group (
        SystemList = { carlton0 = 0, carlton1 = 1 }
        )

        Application uca-ebc (
         User = uca
         StartProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs start"
         StopProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs stop"
         CleanProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs clean"
         MonitorProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs monitor"
         )

        Application uca-ebc-bis (
         User = uca
         StartProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs start bis"
         StopProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs stop bis"
         CleanProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs clean bis"
         MonitorProgram = "/opt/UCA-EBC/bin/uca-ebc-vcs monitor bis"
         )


        // resource dependency tree
        //
        //   group uca-group
        //   {
        //   Application uca-ebc
        //   Application uca-ebc-bis
        //   }
```

Create the virtual IP resource

An IP resource needs mandatory attributes to specify what are the NIC, the IP address and the netmask to use for defining a virtual IP in your subnet.

Here below we are supposing that the configuration is in read-write mode (due to previous `haconf -dump -makero` command)

Run the following commands to:

- Switch the configuration in read-write mode
- Create an HA resource named "uca-ip" to handle the virtual IP address
- Configure the HA resource according your network settings.
- Enable the HA resource
- Validate, apply the configuration and switch it to read-only mode

In order to know what IP address to choose, we suggest to use the same range of addresses as the physical ones. Just choose the right NIC for accessing UCA for EBC server from either a web UI console or from an UCA for EBC Channel Adapter of the NOM platform.

For example, in our configuration, let choose eth0 as the NIC and let suppose we have for both carlton0 and carlton1 addresses like:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:21:5A:45:32:7E
          inet addr:A.B.C.D(*)  Bcast:A.B.C.255  Mask:255.255.255.0
```

(*) A,B and C define your subnet and D differs for carlon0 and carlton1

The we are going to configure a virtual IP address as : A.B.C.V (here below V=253, which is an unregistered physical address):

```
# haconf -makerw

# hares -add uca-ip IP uca-group

# hares -modify uca-ip Device eth0
# hares -modify uca-ip Address A.B.C.253
# hares -modify uca-ip NetMask 255.255.255.0

# hares -modify uca-ip Enabled 1

# haconf -dump -makero
```

Upon successful validation, your VCS configuration will look like:

```
# tail -17 /etc/VRTSvcs/conf/config/main.cf
        IP uca-ip (
         Device = eth0
         Address = "A.B.C.253"
         NetMask = "255.255.255.0"
         )
```

```
// resource dependency tree
//
//   group uca-group
//   {
//   Application uca-ebc
//   IP uca-ip
//   }
```

Create the logical volume and NFS mount resource

It is advised that you configure other resources such as the NFS mount or disk as dependencies for your uca-group.

For example, you could create a resource of type LVMLogicalVolume to add a resource referring to the disk volume and also a resource of type Mount for the mounted point needed to store UCA for EBC data, as referred by the $UCA_EBC_DATA variable.

Those resources are specific to your cluster and therefore are not fully described in this document.

At the end, you are advised to link all resources together so that they are globally used by the service group.

For more in depth configuration of your VCS cluster, refer to *Veritas Cluster Server Administrators Guide*. [R3]

3.4 Monitoring the VCS HA solution

This chapter describes how to monitor UCA for EBC server within the VCS HA solution.

3.4.1 Managing the HA cluster

After installation and configuration the VCS cluster is up and running.

However, here are the main commands to manage it:

Command	Meaning and Parameters
hastart [-stale|-force]	"-stale" instructs the engine to treat the local config as stale "-force" instructs the engine to treat a stale config as a valid one
hasys –force <server_name>	bring the cluster into running mode from a stale state using the configuration file from a particular server
hastop -local	stop the cluster on the local server but leave the application/s running, do not failover the application/s
hastop -local -evacuate	stop cluster on local server but evacuate (failover) the application/s to another node within the cluster
hastop -all -force	stop the cluster on all nodes but leave the application/s running

Command	Meaning and Parameters
hastatus -summary	display cluster summary
hastatus	continually monitor cluster
hasys -display	verify the cluster is operating
haclus -display	information about a cluster
hasys -add <sys>	add a system to the cluster
hasys -delete <sys>	delete a system from the cluster
hasys -modify <sys> <modify options>	Modify a system attributes
hasys -state	list a system state
hasys -force	Force a system to start
hasys -display [-sys]	Display the systems attributes
hasys -list	List all the systems in the cluster
hasys -load <system> <value>	Change the load attribute of a system
hasys -nodeid	Display the value of a systems nodeid (/etc/llthosts)
hasys -freeze [-persistent][-evacuate]	Freeze a system (No offlining system, No groups onlining) Note: main.cf must be in write mode
hasys -unfreeze [-persistent]	Unfreeze a system (reenable groups and resource back online) Note: main.cf must be in write mode

Table 2 - VCS main cluster management commands

3.4.2 Operating the HA service group and/or resource

Once the cluster is well configured, you can operate service group and/or resource, in our case the UCA for EBC server, to bring it offline or online on a specific member of the cluster.

Command	Meaning and Parameters
hagrp -online <group> -sys <sys>	Start a service group and bring its resources online
hagrp -offline <group> -sys <sys>	Stop a service group and takes its resources offline
hagrp -switch <group> to <sys>	Switch a service group from system to another
hagrp -enableresources <group>	Enable all the resources in a group
hagrp -disableresources <group>	Disable all the resources in a group
hagrp -freeze <group> [-persistent]	Freeze a service group (disable onlining and offlining) note: use the following to check "hagrp -display <group> | grep TFrozen"
hagrp -unfreeze <group> [-persistent]	Unfreeze a service group (enable onlining and offlining) note: use the following to check "hagrp -display <group> | grep TFrozen"
haconf -makerw **hagrp -enable <group> [-sys]** **haconf -dump –makero**	Enable a service group. Enabled groups can only be brought online Note to check run the following command "hagrp -display | grep Enabled"
haconf -makerw **hagrp -disable <group> [-sys]** **haconf -dump -makero**	Disable a service group. Stop from bringing online Note to check run the following command "hagrp -display | grep Enabled"
hagrp -flush <group> -sys <system>	Flush a service group and enable corrective action.
hares -online <resource> [-sys]	Online a resource
hares -offline <resource> [-sys]	Offline a resource
hares -state	display the state of a resource(offline, online, etc)
hares -display <resource>	display the parameters of a resource
hares -offprop <resource> -sys <sys>	Offline a resource and propagate the command to its children
hares -probe <resource> -sys <sys>	Cause a resource agent to immediately monitor the resource
hares -clear <resource> [-sys]	Clearing a resource (automatically initiates the onlining)

Table 3 - VCS main service group and resource operations

3.4.3 Practical example

Example with configuration done in previous chapter:

Let's look at states of the service group and resources:

```
# hagrp -state
#Group       Attribute             System     Value
uca-group    State                 carlton0   |OFFLINE|
uca-group    State                 carlton1   |OFFLINE|

# # hares -state
#Resource    Attribute             System     Value
uca-ebc      State                 carlton0   OFFLINE
uca-ebc      State                 carlton1   OFFLINE
uca-ebc-bis  State                 carlton0   OFFLINE
uca-ebc-bis  State                 carlton1   OFFLINE
```

Then disable the "bis" instance (because it is not configured yet):

```
# haconf -makerw
# hares -modify uca-ebc-bis Enabled 0
# haconf -dump –makero
# hares -display | grep -w Enabled | grep -v Type
uca-ebc      Enabled               global     1
uca-ebc-bis  Enabled               global     0
```

Then bring online the service group:

```
# hagrp -online uca-group -any
VCS NOTICE V-16-1-50735 Attempting to online group on system carlton0
# hares -state
#Resource    Attribute             System     Value
uca-ebc      State                 carlton0   ONLINE
uca-ebc      State                 carlton1   OFFLINE
uca-ebc-bis  State                 carlton0   OFFLINE
uca-ebc-bis  State                 carlton1   OFFLINE
[root@carlton0 cluster_server]# hagrp -state
#Group       Attribute             System     Value
uca-group    State                 carlton0   |PARTIAL|
uca-group    State                 carlton1   |OFFLINE|
```

Check that the process is well running on carlton0:

```
# ps -ef | grep UCA-EBC
uca      13165     1  7 18:46 ?        00:00:06 /usr/lib/jvm/java-
1.6.0-openjdk-1.6.0.0.x86_64/bin/java -DUCA-EBC -Duca.instance=default
-Xms1024m -Xmx1024m -XX:PermSize=256m -XX:MaxNewSize=650m -
XX:NewSize=650m -XX:SurvivorRatio=32 -classpath /opt/UCA-
EBC/schemas:/var/opt/UCA-EBC/instances/default/conf:/var/opt/UCA-
EBC/instances/default/deploy:/opt/UCA-EBC/lib/uca-common-
2.0.jar:/opt/UCA-EBC/lib/commons-logging-1.1.1.jar:/opt/UCA-
EBC/lib/jdbcappender-2.1.01.jar:/opt/UCA-EBC/lib/hsqldb-
1.8.0.10.jar:/opt/UCA-EBC/lib/log4j-1.2.16.jar -
Duca.expert.home=/opt/UCA-EBC -Duca.expert.data=/var/opt/UCA-
```

```
EBC/instances/default -Dlog4j.configuration=file:/var/opt/UCA-
EBC/instances/default/conf/uca-ebc-log4j.xml -
Dneo4j.ext.udc.disable=true -
Djava.util.logging.config.file=/var/opt/UCA-
EBC/instances/default/conf/logging.properties
com.hp.uca.common.launch.UcaLauncher
com.hp.uca.expert.engine.Bootstrap start

You can also check that the virtual IP has been well created on
carlton0. This is done by monitoring the new eth0:0 NIC:
# ifconfig eth0:0
eth0:0    Link encap:Ethernet  HWaddr 00:21:5A:45:A8:BA
          inet addr:A.B.C.253  Bcast:A.B.C.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:16 Memory:f6000000-f6012800
```

The web GUI of the UCA for EBC server is now accessible through:

http://A.B.C.253:8888/

The above address is defined on carlton0 only by VCS HA solution.

The virtual IP address is also to be used at UCA for EBC Channel Adapter configuration level in order to have a unique IP address to access UCA for EBC server, whatever host it is running within the cluster.

For example, this would give for default instance:

```
<bean id="activemq-uca-ebc"
class="org.apache.activemq.camel.component.ActiveMQComponent">
  <property name="brokerURL" value="tcp://A.B.C.253:61666"/> </bean>
```

Now let's simulate a failover.

As an example, kill the process on carlton0. Wait a few seconds for the HA mechanism to trigger the failover on carlton1.

Then check the state of the HA resource:

```
# hares -state
#Resource    Attribute             System     Value
uca-ebc      State                 carlton0   FAULTED
uca-ebc      State                 carlton1   ONLINE
uca-ebc-bis  State                 carlton0   OFFLINE
uca-ebc-bis  State                 carlton1   OFFLINE
```

Listing 1 -checking state of HA resources

You will notice that the resource has passed to FAULTED state on carlton0 and that has been successfully restarted on carlton1.

You can also notice that the virtual IP has also been switched from carlton0 to carlton1, and from now on a call to ***http://A.B.C.253:8888/*** will be forwarded to carlton1 (the new active member).

For sanity of your platform, just declare that carlton0 is no more faulty (let's suppose we have fix an hypothetic problem). To do that, clear the resource:

```
# hares -clear uca-ebc -sys carlton0
```

3.4.4 Troubleshooting

Look at file /var/VRTSvcs/log/engine_A.log for logs generated by the VCS engine.

For example, with previous commands the log file should have logs like:

When bringing online the service group, the logs of the startup:

```
2012/12/12 18:46:21 VCS NOTICE V-16-1-10301 Initiating Online of
Resource uca-ebc (Owner: Unspecified, Group: uca-group) on System
carlton0
2012/12/12 18:46:21 VCS INFO V-16-10031-504 (carlton0)
Application:uca-ebc:online:Executed /opt/UCA-EBC/bin/uca-ebc-vcs as
user uca2012/12/12 18:46:23 VCS INFO V-16-1-10298 Resource uca-ebc
(Owner: Unspecified, Group: uca-group) is online on carlton0 (VCS
initiated)
```

After the killing of the process on carlton0, the logs of the detection:

```
2012/12/12 18:52:24 VCS ERROR V-16-2-13067 (carlton0) Agent is calling
clean for resource(uca-ebc) because the resource became OFFLINE
unexpectedly, on its own.
2012/12/12 18:52:24 VCS INFO V-16-10031-504 (carlton0)
Application:uca-ebc:clean:Executed /opt/UCA-EBC/bin/uca-ebc-vcs as
user uca
2012/12/12 18:52:35 VCS INFO V-16-2-13068 (carlton0) Resource(uca-ebc)
- clean completed successfully.
2012/12/12 18:52:36 VCS INFO V-16-1-10307 Resource uca-ebc (Owner:
Unspecified, Group: uca-group) is offline on carlton0 (Not initiated
by VCS)
2012/12/12 18:52:36 VCS ERROR V-16-1-10205 Group uca-group is faulted
on system carlton0
2012/12/12 18:52:36 VCS NOTICE V-16-1-10446 Group uca-group is offline
on system carlton0
2012/12/12 18:52:36 VCS INFO V-16-1-10493 Evaluating carlton0 as
potential target node for group uca-group
2012/12/12 18:52:36 VCS INFO V-16-1-50010 Group uca-group is online or
faulted on system carlton0
2012/12/12 18:52:36 VCS INFO V-16-1-10493 Evaluating carlton1 as
potential target node for group uca-group
2012/12/12 18:52:36 VCS NOTICE V-16-1-10301 Initiating Online of
Resource uca-ebc (Owner: Unspecified, Group: uca-group) on System
carlton1
```

When failover has occurred, the logs of the startup on other member:

```
2012/12/12 18:52:36 VCS INFO V-16-10031-504 (carlton1)
Application:uca-ebc:online:Executed /opt/UCA-EBC/bin/uca-ebc-vcs as
user uca
```

```
2012/12/12 18:53:38 VCS INFO V-16-1-10298 Resource uca-ebc (Owner:
Unspecified, Group: uca-group) is online on carlton1 (VCS initiated)
```

Chapter 4

The UCA for EBC HA solution based on RHEL cluster

4.1 Introducing the Red Hat Enterprise Linux High Availability Add-On

The actual name of the software, in a RHEL cluster, that manages the High Availability is called the `RHEL High Availability Add-On`.

Red Hat's High Availability Add-On provides on-demand failover services between nodes within a cluster, making applications highly available. The High Availability Add-On supports up to 16 nodes.

When using the High Availability Add-On, a highly available service can fail over from one node to another with no apparent interruption to cluster clients.

The High Availability Add-On also ensures data integrity when one cluster node takes over control of a service from another cluster node.

It achieves this by promptly evicting nodes from the cluster that are deemed to be faulty using a method called "fencing" that prevents data corruption.

The complete High Availability Add-On datasheet is available at:
http://www.redhat.com/rhecm/rest-rhecm/jcr/repository/collaboration/jcr:system/jcr:versionStorage/704a02ed0a052601101495b1f5af3e4c/1/jcr:frozenNode/rh:resourceFile

4.2 Installing the Red Hat Enterprise Linux High Availability Add-On

The RHEL cluster management software is called "**Conga**".
Conga is made of agent software, called "**ricci**", and running on the nodes where UCA for EBC applications run.
Conga is also made of a centralized cluster configuration & management server, called "**luci**".

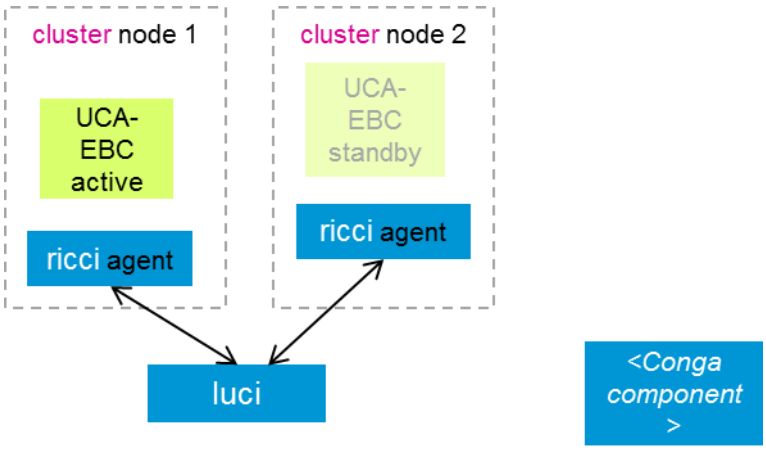

Figure 2 - luci & ricci Conga components in a two-node cluster

Installation procedure

Install RHEL 6.4 or newer on all the nodes of your cluster

Install group packages for necessary cluster and cluster management software on cluster nodes. The "High Availability" group package is mandatory. The "Resilient Storage" group package will be needed if you want your UCA-EBC service to be based on a resilient shared directory.
We advise to install the necessary software one at a time and reboot. Repeat these commands on each node of the cluster.

```
[root@nodeN ~]# yum -y groupinstall "High Availability" "Resilient Storage"
```

On the node chosen to become the management node of the cluster, install the luci component. In our example the MgtNode is node1.

```
[root@MgtNode ~]# yum -y install luci ccs
```

ccs is the command line set of tools allowing to do most of what can be done with **luci**

4.3 Configuring the Red Hat Enterprise Linux High Availability Add-On

4.3.1 Configuring the iptables Firewall to Allow Cluster Components

To allow the nodes in a cluster to communicate with each other, you must enable the IP ports assigned to certain Red Hat High Availability Add-On components

Listed below are iptable rules for enabling IP ports needed by RHEL 6 High Availability Add-on. Please note that these examples use 192.168.1.0/24 as a subnet, but you will need to replace 192.168.1.0/24 with the appropriate subnet.

For **cman** (cluster manager):

```
#  iptables -I INPUT -m state --state NEW -m multiport -p udp -s
192.168.1.0/24 -d 192.168.1.0/24 --dports 5404,5405
-j ACCEPT

# iptables -I INPUT -m addrtype --dst-type MULTICAST -m state --state
NEW -m multiport -p udp -s 192.168.1.0/24
--dports 5404,5405 -j ACCEPT
```

For **dlm** (Distributed Lock Manager):

```
# iptables -I INPUT -m state --state NEW -p tcp
-s 192.168.1.0/24 -d 192.168.1.0/24 --dport 21064 -j ACCEPT
```

For **ricci** (part of Conga remote agent):

```
#  iptables -I INPUT -m state --state NEW -p tcp
-s 192.168.1.0/24 -d 192.168.1.0/24 --dport 11111 -j ACCEPT
```

For **modclusterd** (part of Conga remote agent):

```
#  iptables -I INPUT -m state --state NEW -p tcp
-s 192.168.1.0/24 -d 192.168.1.0/24 --dport 16851 -j ACCEPT
```

For **luci** (Conga User Interface server):

```
#  iptables -I INPUT -m state --state NEW -p tcp
-s 192.168.1.0/24 -d 192.168.1.0/24 --dport 8084 -j ACCEPT
```

For **igmp** (Conga User Interface server):

```
#  iptables -I INPUT -p igmp -j ACCEPT
```

After executing these commands, run the following command to save the current configuration for the changes to be persistent during reboot

```
#  service iptables save ; service iptables restart
```

4.3.2 Considerations about SELinux

The High Availability Add-On for Red Hat Enterprise Linux 6 supports SELinux in the enforcing state with the SELinux policy type set to targeted.

When using SELinux with the High Availability Add-On in a VM environment, you should ensure that the SELinux boolean fenced_can_network_connect is persistently set to on. This allows the fence_xvm fencing agent to work properly, enabling the system to fence virtual machine.

4.3.3 Considerations about ricci

It is necessary that **ricci** is running in each cluster node to be able to propagate updated cluster configuration.
You can start ricci by using `service ricci start` or by enabling it to start at boot time via **chkconfig**

4.4 Creating a Highly Available UCA-EBC service

4.4.1 Creating a cluster

Start luci using service luci start (or enable it to start at boot time via chkconfig)

```
[root@MgtNode ~]#  service luci start

Starting luci: generating https SSL certificates... done

[ OK ]
```

The URL syntax for the luci server is

https://luci_server_hostname:luci_server_port. The default value of luci_server_port is 8084.

Point your web browser to **https://< MgtNode>:8084** to access luci

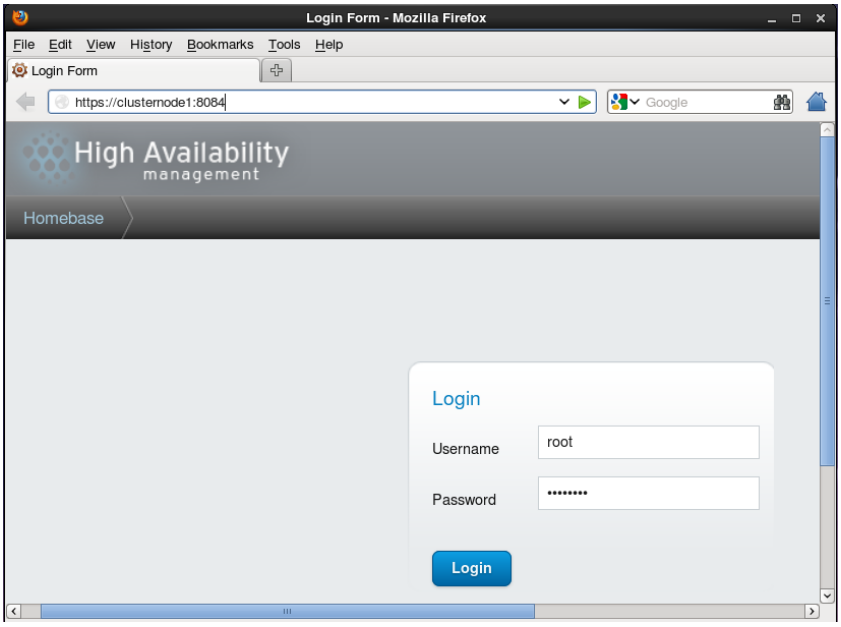

Figure 3 - luci homepage URL

The root user on a system running luci can control access to the various luci components by setting permissions for the individual users on a system.

Click Manage Clusters from the menu on the left side of the luci Homebase page

The Clusters screen appears,

Click Create. The Create New Cluster dialog box appears, as shown in Figure 4 below

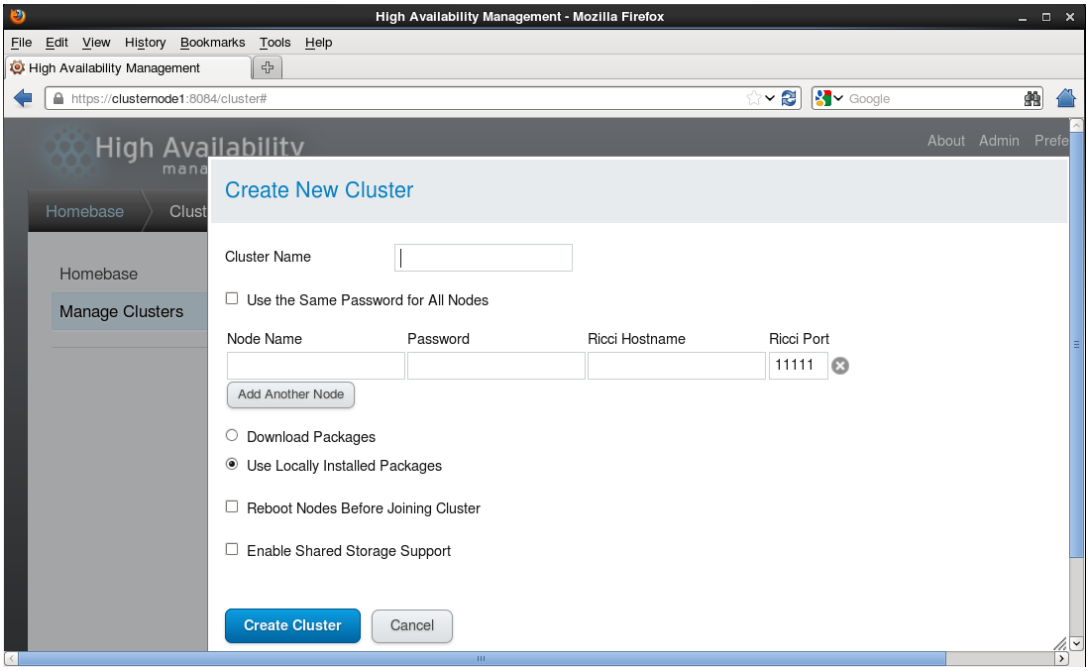

Figure 4 - luci cluster creation page

After adding your nodes and clicking the Create Cluster button, your cluster should look like the example in the below Figure 5

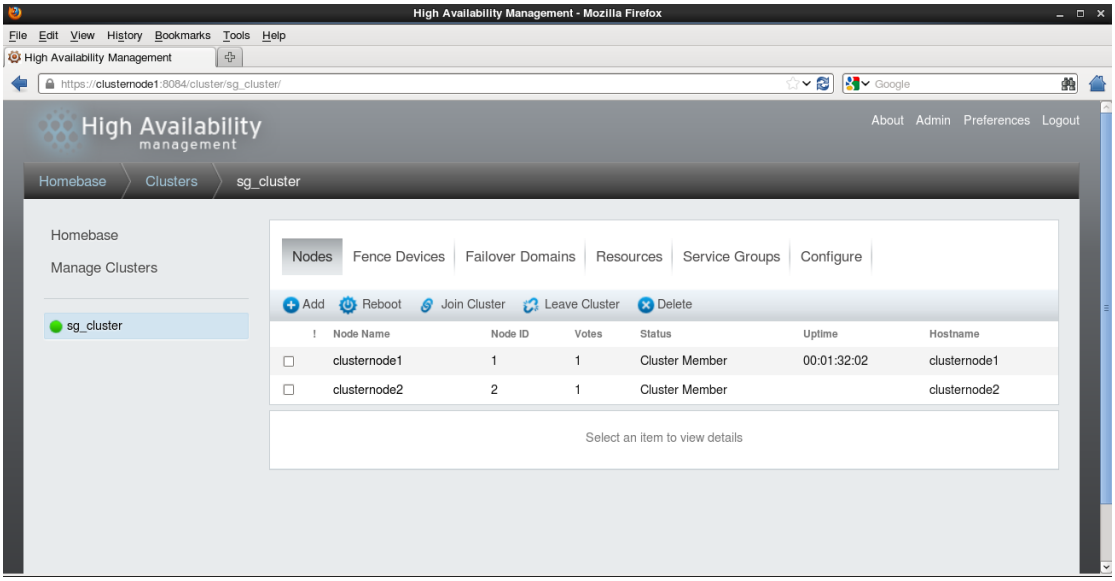

Figure 5 - viewing the created cluster in luci cluster management page

Your cluster structure is now created.

4.4.2 Configuring Fencing

Fencing consists in putting in quarantine a node of the cluster that is not functionning properly in order to protect shared resources

Fencing may either disable a failing node, or disallow shared storage access, therefore ensuring data integrity.

The configuration of fencing in a RHEL cluster can be done in two steps.
First step is about configuring fence devices.
Second step is about configuring fencing for the cluster members.

Configuring fence devices

Configuring fence devices consists of creating, updating, and deleting fence devices for the cluster. You must configure the fence devices in a cluster before you can configure fencing for the nodes in the cluster.

Creating a fence device consists of selecting a fence device type and entering parameters for that fence device (for example, name, IP address, login, and password).

For information about Updating or Deleting a fence device, refer to [R5] RHEL 6 - Cluster Administration

From the cluster-specific page, you can configure fence devices for that cluster by clicking on Fence Devices along the top of the cluster display. This displays the fence devices for the cluster and displays the menu items for fence device configuration: Add and Delete.

To create a fence device, follow these steps:

1. From the Fence Devices configuration page, click Add. Clicking Add displays the Add Fence Device (Instance) dialog box. From this dialog box, select the type of fence device to configure.

2. Specify the information in the Add Fence Device (Instance) dialog box according to the type of fence device.

Refer to[R5] RHEL 6 - Cluster Administration to see the list of possible fence devices

In our example we have chosen HP ILO Devices (one ILO device per cluster member)

3. Click Submit

The result in our example looks like the Figure 6 below

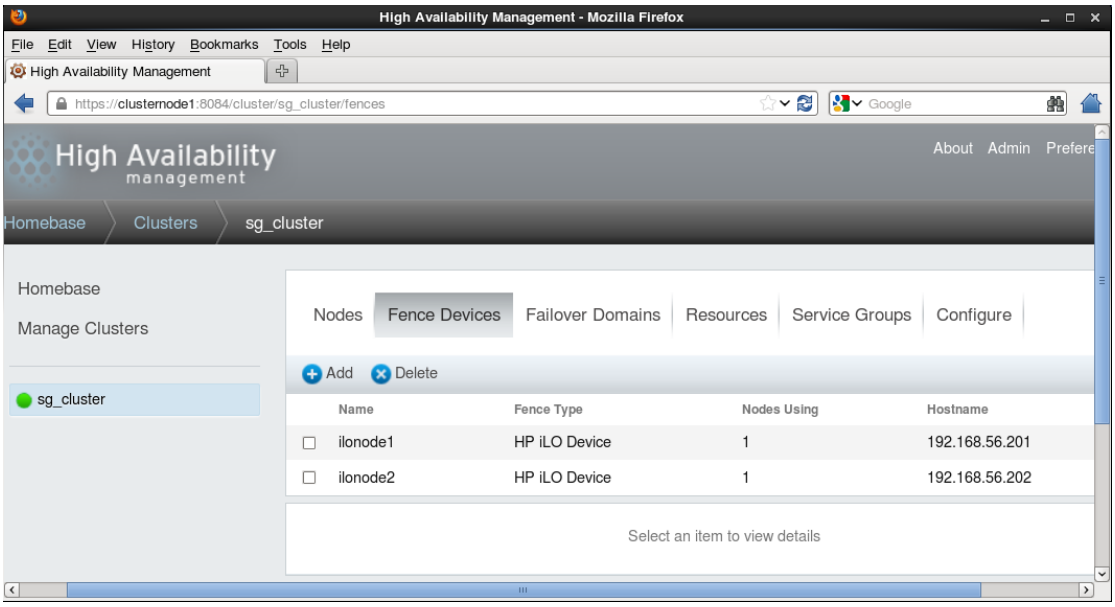

Figure 6 - Fence devices management page

Configuring fencing for the cluster nodes

Once you have completed the initial steps of creating a cluster and creating fence devices, you need to configure fencing for the cluster nodes. To configure fencing for the nodes after creating a new cluster and configuring the fencing devices for the cluster, follow the steps in this section. Note that you must configure fencing for each node in the cluster.

Use the following procedure to configure a node with a single fence device.

1. From the cluster-specific page, you can configure fencing for the nodes in the cluster by clicking on Nodes along the top of the cluster display. This displays the nodes that constitute the cluster.

This is also the default page that appears when you click on the cluster name beneath Manage Clusters from the menu on the left side of the luci Homebase page.

2. Click on a node name. Clickin g a link for a node causes a page to be displayed for that link showing how that node is configured.

3. On the node-specific page, under Fence Devices, click Add Fence Method. This displays the Add Fence Method to Node dialog box.

4. Enter a Method Name for the fencing method that you are configuring for this node. This is an arbitrary name that will be used by Red Hat High Availability Add-On; it is not the same as the DNS name for the device.

5. Click Submit. This displays the node-specific screen that now displays the method you have just added under Fence Devices.

6. Configure a fence instance for this method by clicking the Add Fence Instance button that appears beneath the fence method. This displays the Add Fence Device (Instance) drop-down menu from which you can select a fence device you have previously configured.

7. Select a fence device for this method.

In our example the result looks like what is shown in Figure 7 below

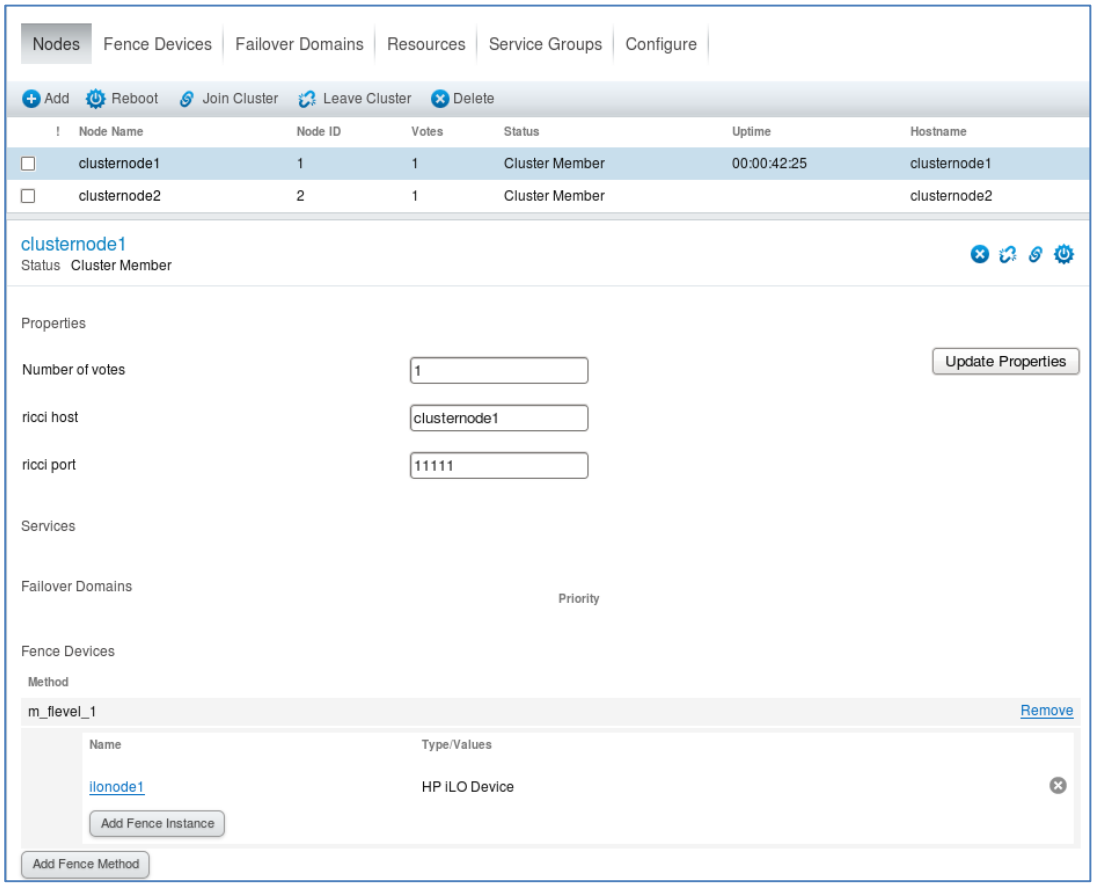

Figure 7 - Associating a fence device to a cluster member

Note that it is recommended to configure multiple fencing mechanisms for each node. A fencing device can fail. However configuring multiple fence devices is not described in this guide.
Please refer to [R5] RHEL 6 - Cluster Administration

4.4.3 Configuring a Failover Domain

A failover domain is a named subset of cluster nodes that are eligible to run a cluster service in the event of a node failure.

In this guide no failover have been configured since the guide is based on a two-node cluster only.

4.4.4 Creating the UCA for EBC service

We can now declare our UCA for EBC service. It will be created in the cluster as a service group. And then in next paragraph, we will see how to add resources to this UCA for EBC service, so that it should become a HA service.

From the cluster-specific page, you can add the UCA-EBC service to that cluster by clicking on Service Groups along the top of the cluster display.

2. Click Add. This displays the Add Service Group to Cluster dialog box.

3. On the Add Service Group to Cluster dialog box, at the Service Name text box, type the name of the UCA for EBC service, e.g. **UCA-EBC**.

4. Check the Automatically Start This Service checkbox if you want the UCA-EBC service to start automatically when the cluster is started.

5. Check the Run Exclusive checkbox to set a policy wherein the service only runs on nodes that have no other services running on them.

6. If you have configured failover domains for the cluster, you can use the drop-down menu of the Failover Domain parameter to select a failover domain for this service.

7. Use the Recovery Policy drop-down box to select a recovery policy for the service. We recommend to use the option Relocate, and therefore to ignore the restart options.

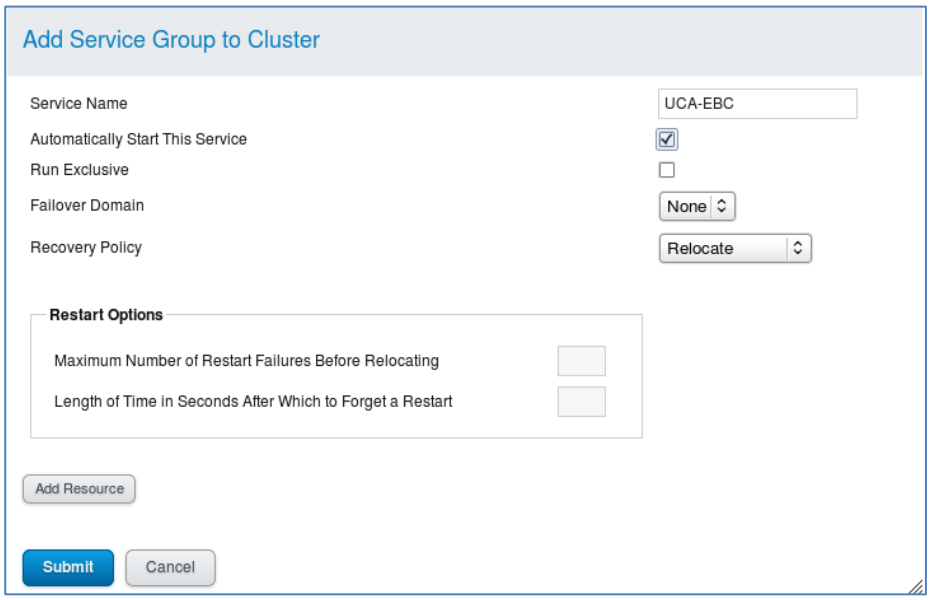

Figure 8 - Creating the UCA-EBC Service Group in the cluster

You may press the Submit now and add the resources later as explained in the following paragraph.

4.4.5 Adding the resources for the UCA-EBC HA service

We have the possibility to add resources at the cluster level, where they will be visible and useable by any service in the cluster; or to add resources at the service level, where they will exclusively be available for the service in question.

In this guide the UCA-EBC control script and the virtual IP address are defined at UCA-EBC service level, whereas the HA LVM and File System are defined at cluster level.

UCA-EBC script

To start, stop and get the status of the UCA-EBC service, we rely on the script `uca-ebc-rhelcluster`

In the Service Groups tab, click on the UCA-EBC service, and then on the Add Resource button.

Then from the dropdown list choose Script

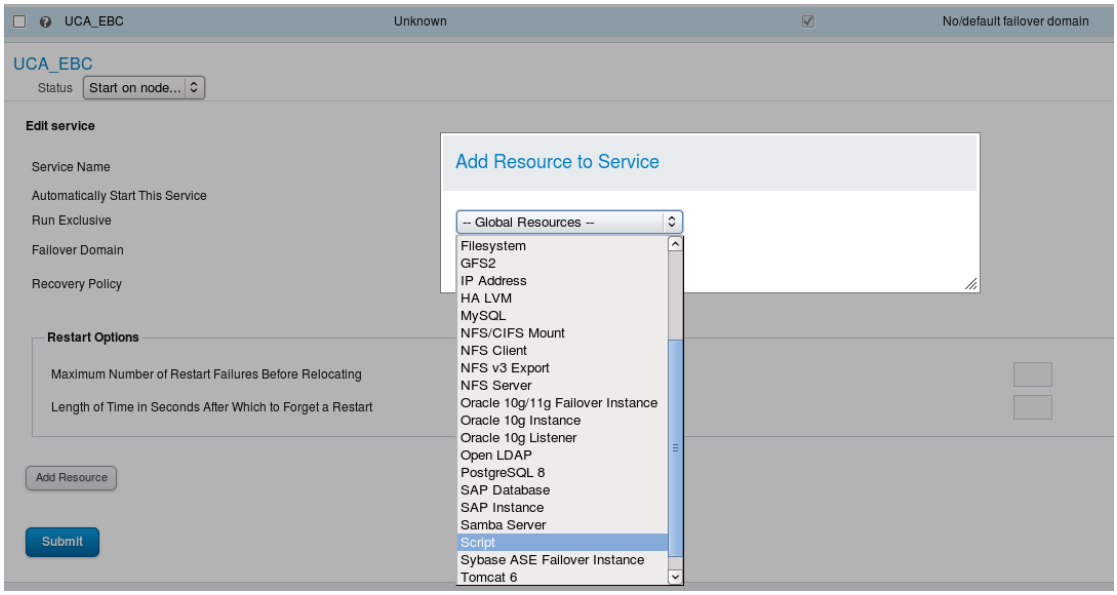

Figure 9 - Adding a script resource to the UCA-EBC service

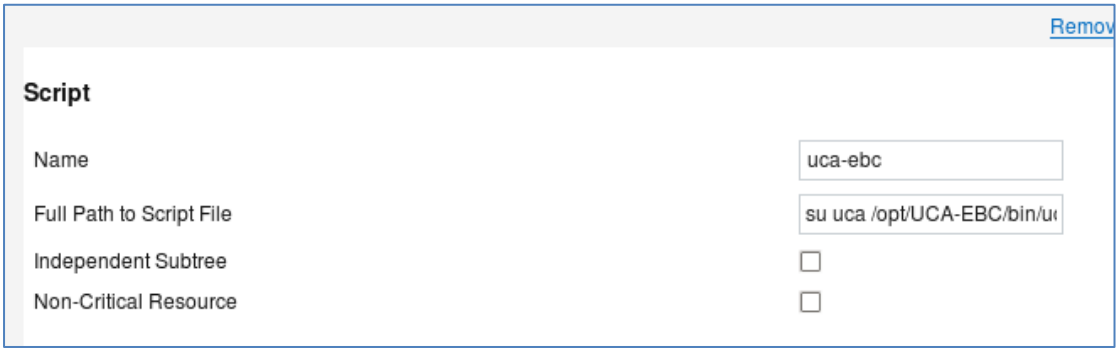

Figure 10 - Specifying the location of the UCA-EBC control script file

In the Full Path to Script File put
su uca /opt/UCA-EBC/bin/uca-ebc-rhelcluster

It is important to run the script as **uca** user

IP address

As stated before in paragraph 3.3.3, we need the UCA-EBC HA service to be always accessed with the same virtual IP address by other components in the solution such as the Open Mediation.

In the Service Groups tab, click on the UCA-EBC service, and then on the Add Resource button.

Then from the dropdown list choose IP Address

Luci Field	Description
IP Address, Netmask Bits	The IP address (and, optionally, netmask bits) for the resource. Netmask bits, or network prefix length, may come after the address itself with a slash as a separator, complying with CIDR notation (for example, 10.1.1.1/8). This is a virtual IP address. IPv4 and IPv6 addresses are supported, as is NIC link monitoring for each IP address.
Monitor Link	Enabling this causes the status check to fail if the link on the NIC to which this IP address is bound is not present.
Disable Updates to Static Routes	Disable updating of routing using RDISC protocol.
Number of Seconds to Sleep After Removing an IP Address	Specifies the amount of time (in seconds) to sleep.

Table 4 - RHEL cluster IP address resource properties

HA LVM

The choice retained in this guide for the shared storage of the UCA-EBC HA service is to use HA LVM.

As stated in the Appendix F of the [R5] RHEL 6 - Cluster Administration

If the applications run optimally in active/passive (failover) configurations where only a single node that accesses the storage is active at any one time, you should use High Availability Logical Volume Management agents (HA-LVM).

To set up HA-LVM failover, perform the following steps:

Ensure that your system is configured to support CLVM, which requires the following:

The High Availability Add-On and Resilient Storage Add-On are installed, including the the cmirror package if the CLVM logical volumes are to be mirrored.

The locking_type parameter in the global section of the /etc/lvm/lvm.conf file is set to the value '3'.

The High Availability Add-On and Resilient Storage Add-On software, including the clvmd daemon, must be running. For CLVM mirroring, the cmirrord service must be started as well.

Create the logical volume and file system using standard LVM and file system commands, as in the following example.

```
[root@nodeN ~]# pvcreate /dev/sd[cde]1

[root@nodeN ~]# vgcreate -cy shared_vg /dev/sd[cde]1
```

```
[root@nodeN ~]# lvcreate -L 10G -n ha_lv shared_vg

[root@nodeN ~]# mkfs.ext4 /dev/shared_vg/ha_lv

[root@nodeN ~]# lvchange -an shared_vg/ha_lv

__
```

For information on creating LVM logical volumes, refer to *Logical Volume Manager Administration*.

_

Use the luci interface to add the newly created lvm as a resource of the cluster, as shown in Figure 11 below.

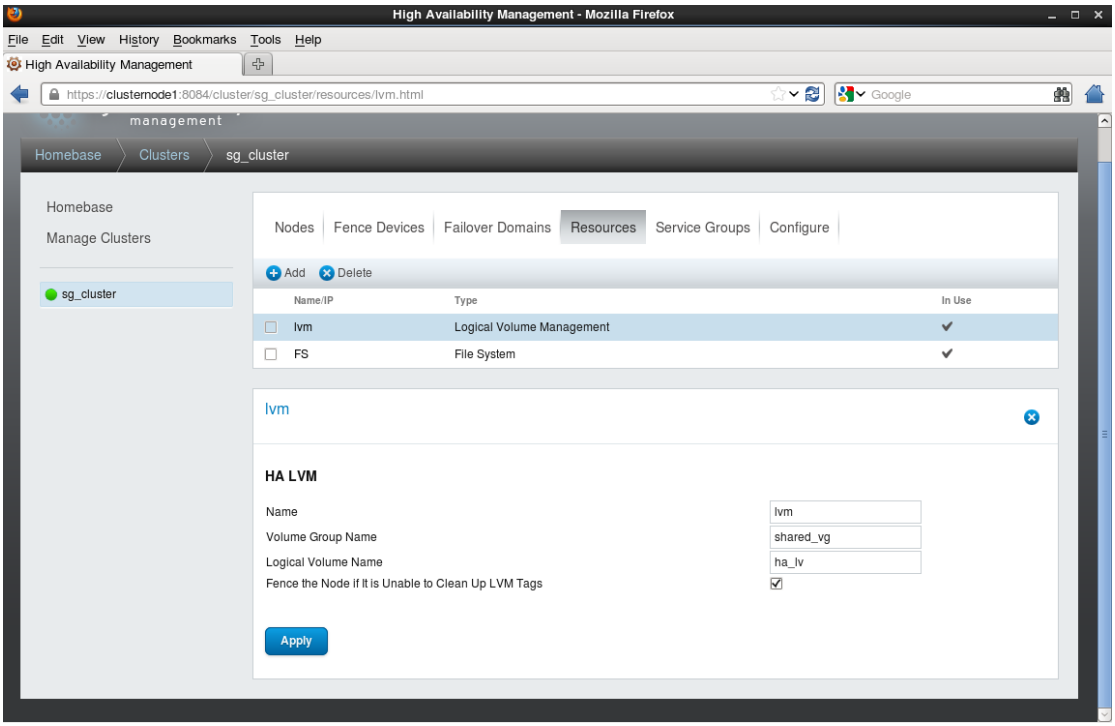

Figure 11 - Adding the HA LVM resource to the cluster

And then add this resource to the UCA-EBC service group, in the Service Groups tab

Filesystem

Following the creation of the HA LVM, create a Filesystem resource on top of the lvm, as shown below.

Note that the mount point chosen (/var/shared/UCA_EBC) corresponds exactly to the data directory selected when installing UCA for EBC application on each of the cluster nodes. (see Chapter 2)

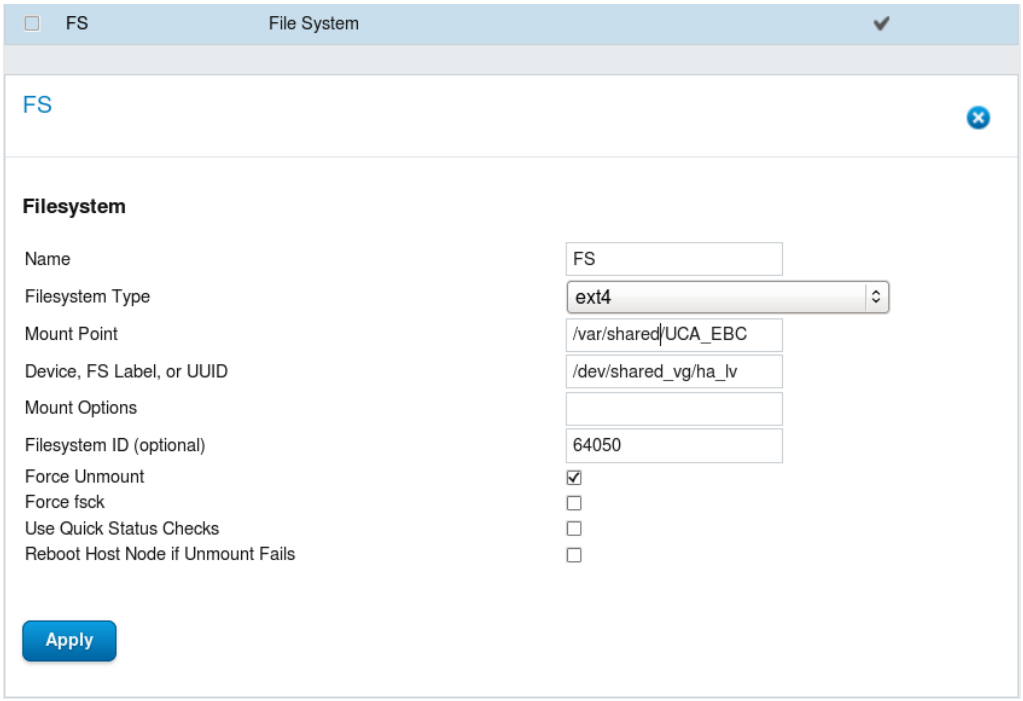

Figure 12 - Creating the File System for the shared UCA-EBC data

And then add this resource to the UCA-EBC service group, in the Service Groups tab.

4.5 Managing the cluster

Refer to **[R5]** RHEL 6 - Cluster Administration **Chapter 4. Managing Red Hat High Availability Add-On With Conga**

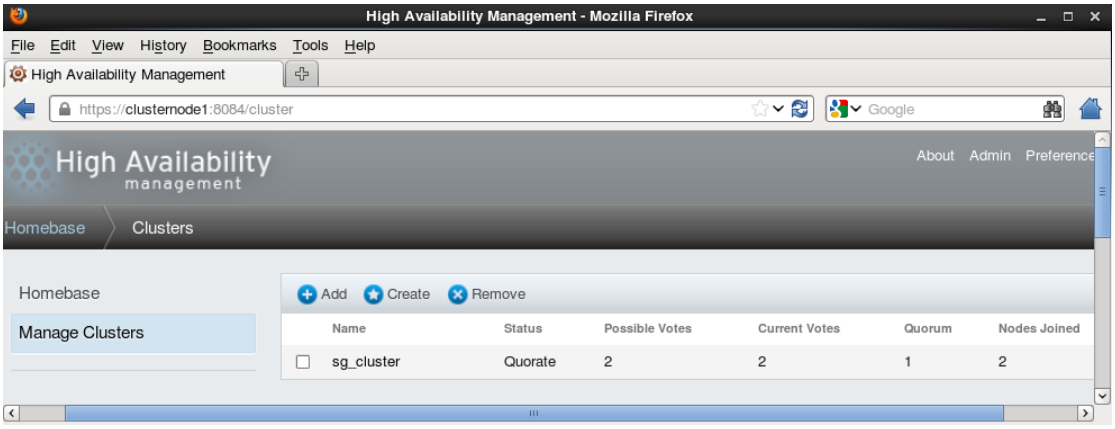

Figure 13 - Cluster Management page in luci

4.6 Managing the HA UCA for EBC service

You can manage your UCA-EBC HA service in the luci interface as shown in the Figure 14 below

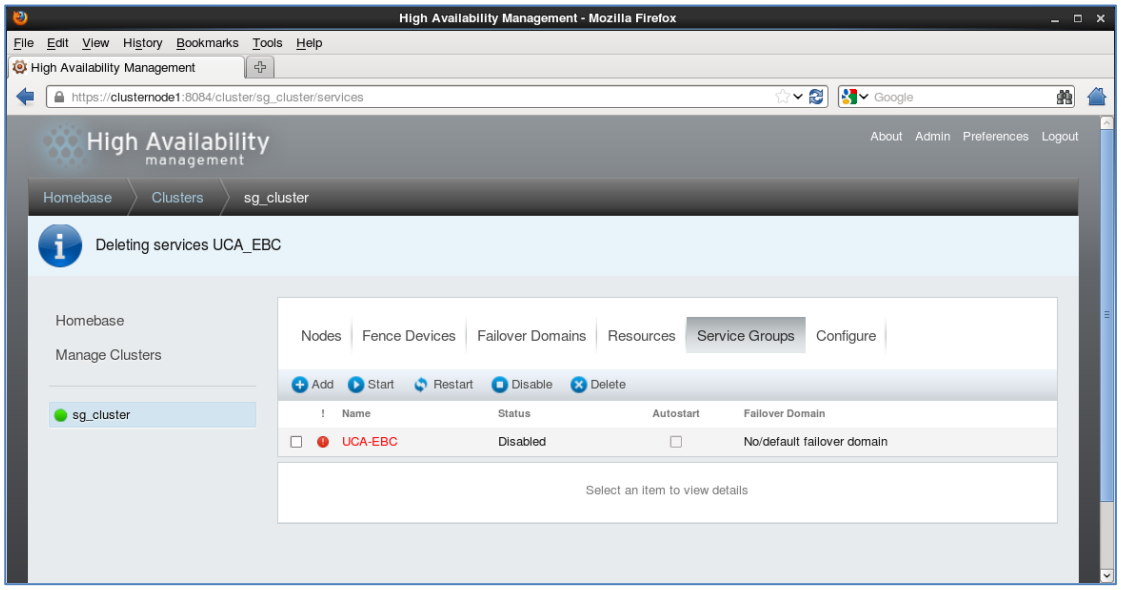

Figure 14 - UCA-EBC HA service control page in luci

4.7 Troubleshooting

View the logs of execution of the service script under the local /var/tmp/

Those logs have not been put on the normal UCA_EBC_DATA log directory for troubleshooting reasons. Since the UCA_EBC_DATA directory is a shared directory, and is sometimes mounted on one node, sometimes on the other node, in case of problem, it can be difficult to find the log file.

The log files corresponding to

Start of the UCA-EBC service : **rhelcluster-uca-service-start.out**

Stop of the UCA-EBC service : **rhelcluster-uca-service-stop.out**

Status of the UCA-EBC service : **rhelcluster-uca-service-status.out**

```
[root@nodeN ~]# ls -l /var/tmp/

-rw-r--r--. 1 root     root     99818 Dec  4 17:32 rhelcluster-uca-
service-start.out
-rw-r--r--. 1 root     root       236 Dec  4 17:32 rhelcluster-uca-
service-status.out
-rw-r--r--. 1 root     root       323 Dec 22 15:16 rhelcluster-uca-
service-stop.out
```

You can check the normal behavior in the /var/log/messages file

When starting the UCA-EBC service on one node, you should see logs similar to the ones below

/var/log/messages

```
[root@nodeWhereUCAWasStarted ~]# tail –f /var/log/messages

11:43:59 clusternode2 rgmanager[2746]: Starting disabled service
service:UCA-EBC
11:44:00 clusternode2 rgmanager[4442]: [fs] mounting /dev/dm-0 on
/usr/share/UCA_EBC_DATA
11:44:00 clusternode2 rgmanager[4464]: [fs] mount -t ext4  /dev/dm-0
/usr/share/UCA_EBC_DATA
11:44:00 clusternode2 kernel: EXT4-fs (dm-0): warning: maximal mount
count reached, running e2fsck is recommended
11:44:00 clusternode2 kernel: EXT4-fs (dm-0): recovery complete
11:44:00 clusternode2 kernel: EXT4-fs (dm-0): mounted filesystem with
ordered data mode. Opts:
11:44:00 clusternode2 rgmanager[4549]: [ip] Adding IPv4 address
192.168.56.111/24 to eth0
11:44:02 clusternode2 avahi-daemon[2065]: Registering new address
record for 192.168.56.111 on eth0.IPv4.
11:44:03 clusternode2 rgmanager[4634]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster start
11:44:04 clusternode2 rgmanager[2746]: Service service:UCA-EBC started
```

When the heartbeat of the service on one node works well, you should see the logs below

```
[root@nodeWhereUCAWasStarted ~]# tail –f /var/log/messages

11:44:34 clusternode2 rgmanager[4957]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster status

11:45:14 clusternode2 rgmanager[5552]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster status

11:45:54 clusternode2 rgmanager[6005]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster status

11:46:34 clusternode2 rgmanager[6664]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster status
```

When the node hosting the service is stopped or failed, logs will be similar to these ones below

```
[root@nodeWhereUCAWasStopped ~]# tail –f /var/log/messages

11:52:00 clusternode2 rgmanager[2746]: Stopping service service:UCA-
EBC
11:52:01 clusternode2 rgmanager[11417]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster stop
11:52:03 clusternode2 rgmanager[11521]: [ip] Removing IPv4 address
192.168.56.111/24 from eth0
11:52:03 clusternode2 avahi-daemon[2065]: Withdrawing address record
for 192.168.56.111 on eth0.
11:52:13 clusternode2 rgmanager[11627]: [fs] unmounting
/usr/share/UCA_EBC_DATA
```

```
11:52:13 clusternode2 rgmanager[2746]: Service service:UCA-EBC is
stopped
```

The logs on the node where UCA-EBC service is relocated will look like these ones below.

```
[root@nodeWhereUCAWasStarted ~]# tail -f /var/log/messages

11:52:15 clusternode1 rgmanager[1278]: Starting disabled service
service:UCA-EBC
11:52:16 clusternode1 rgmanager[1825]: [fs] mounting /dev/dm-0 on
/usr/share/UCA_EBC_DATA
11:52:16 clusternode2 rgmanager[2019]: [fs] mount -t ext4  /dev/dm-0
/usr/share/UCA_EBC_DATA
11:52:16 clusternode1 kernel: EXT4-fs (dm-0): warning: maximal mount
count reached, running e2fsck is recommended
11:52:22 clusternode1 kernel: EXT4-fs (dm-0): recovery complete
11:52:22 clusternode2 kernel: EXT4-fs (dm-0): mounted filesystem with
ordered data mode. Opts:
11:52:26 clusternode1 rgmanager[3134]: [ip] Adding IPv4 address
192.168.56.110/24 to eth0
11:52:26 clusternode1 avahi-daemon[2045]: Registering new address
record for 192.168.56.110 on eth0.IPv4.
11:52:26 clusternode1 rgmanager[3256]: [script] Executing /opt/UCA-
EBC/bin/uca-ebc-rhelcluster start
11:52:30 clusternode1 rgmanager[2078]: Service service:UCA-EBC started
```

4.8 Configuration example

The HA UCA-EBC service running on a two-node cluster, that was taken as an example in this guide has the following configuration /etc/cluster/cluster.conf

```
<?xml version="1.0"?>
<cluster config_version="35" name="sg_cluster">
        <clusternodes>
                <clusternode name="clusternode1" nodeid="1">
                        <fence>
                                <method name="m_flevel_1">
                                        <device name="ilonode1"/>
                                </method>
                        </fence>
                </clusternode>
                <clusternode name="clusternode2" nodeid="2">
                        <fence>
                                <method name="m_flevel_2">
                                        <device name="ilonode2"/>
                                </method>
                        </fence>
                </clusternode>
        </clusternodes>
        <cman expected_votes="1" two_node="1"/>
```

```
        <rm>
            <resources>
                        <lvm lv_name="ha_lv" name="lvm"
self_fence="on" vg_name="shared_vg"/>
                        <fs device="/dev/shared_vg/ha_lv"
force_unmount="on" fsid="64050" fstype="ext4"
mountpoint="/usr/share/UCA_EBC_DATA" name="FS"/>
            </resources>
            <service autostart="0" name="UCA-EBC" recovery="relocate">
                        <script file="/opt/UCA-EBC/bin/uca-ebc-
rhelcluster" name="uca-ebc"/>
                        <ip address="192.168.56.111" sleeptime="10"/>
                        <lvm ref="lvm"/>
                        <fs ref="FS"/>
            </service>
        </rm>
        <fencedevices>
                <fencedevice agent="fence_ilo" ipaddr="192.168.56.201"
login="root" name="ilonode1" passwd="Plusvite"/>
                <fencedevice agent="fence_ilo" ipaddr="192.168.56.202"
login="root" name="ilonode2" passwd="Plusvite"/>
        </fencedevices>
</cluster>
```

For testing purposes, the pd-example UCA-EBC Problem Detection Value Pack has been deployed on the UCA-EBC HA service.

Note that it is only needed to deploy a VP on the node where the active UCA for EBC application runs.

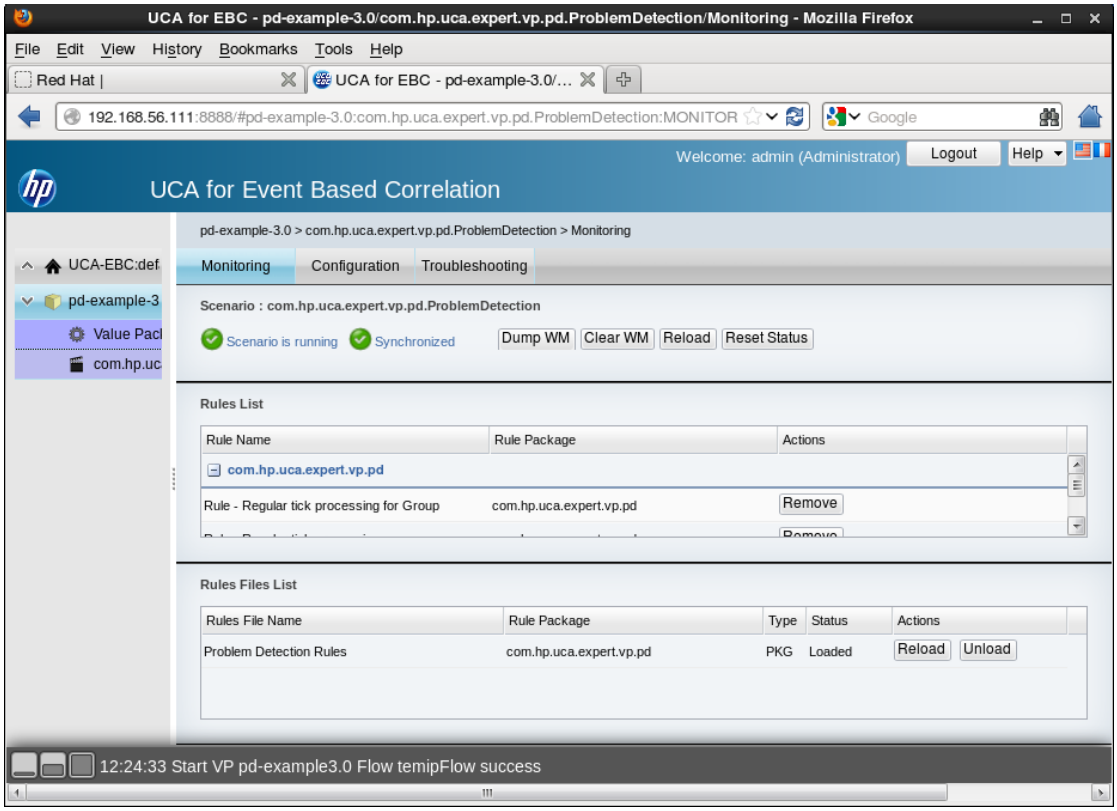

Figure 15 - HA UCA-EBC GUI showing the example Problem Detection VP

Chapter 5
Glossary

UCA: Unified Correlation Analyzer

EBC: Event Based Correlation

VCS: Veritas Cluster Server

JDK: Java Development Kit

JRE: Java Runtime Environment

HA: High Availability