hp Unified Correlation Analyzer

Unified Correlation Analyzer
for
Event Based Correlation

Version 3.0

Clustering and High-Availability Guide

Edition: 1.0

For Linux (RHEL 6.2 (Santiago)©) x86_64 Operating Systems

June 2013

Legal Notices

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.

License Requirement and U.S. Government Legend

Confidential computer software. Valid license from HP 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.

Copyright Notices

Trademark Notices

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.

Contents

Figures

No table of figures entries found.

Tables

Preface

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

Product Version: 3.0

Kit Version: V3.0

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 V3.0	• Red Hat Enterprise Linux Server release 6.2 (Santiago)

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:

References

[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

Support

Please visit our HP Software Support Online Web site at www.hp.com/go/hpsoftwaresupport for contact information, and details about HP 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 clustering procedures for high-availability (HA) setup of the UCA for EBC product within a cluster.

The term cluster refers to multiple independent systems connected into a management framework.

Currently, the Veritas Cluster Server (VCS) from Symantec company is the only high-availability solution described in this document. It does not mean that UCA for EBC product is supported only on VCS, but solely that other solutions are not yet fully described.

1.1 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).

1.1.1 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.

1.1.2 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]

1.2 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 V3.0 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].

This guide provides instructions on how to configure UCA for EBC server to be managed for HA by VCS solution, as an HA resource. UCA for EBC server will be managed through the VCS Application Agent.

However, please refer to the *Unified Correlation Analyzer for Event Based Correlation Installation and Configuration Guide* [R1] for a better understanding on how to install UCA for EBC server in order to support HA mode.

For more information about VCS Application Agent, refer to *Veritas Cluster Server Bundled Agents Reference Guide* [R3].

Chapter 2

Installation and prerequisites

This chapter describes the installation steps to be performed to install UCA for EBC product within VCS cluster environment.

2.1 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 trialware 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

2.2 Installing UCA for EBC server

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

uca-ebc-server-kit-3.0-linux.tar

For detailed installations instructions and license setup, refer to the *Unified Correlation Analyzer for Event Based Correlation Installation Guide* [R1].

UCA for EBC comes with 90 days of free usage.

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-1.6.0-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 1.6 JRE (Java Runtime Environment) and have JAVA_HOME defined accordingly.

Chapter 3

Configuration of a VCS HA solution

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

3.1 Setting up an 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
```

3.2 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 Creating HA resources for UCA-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 V3.0of 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**.

3.3.1 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
        //   }
```

3.3.2 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
        //   }
```

3.3.3 Create the logical volume and NFS mount resources

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]

Chapter 4

Monitoring the VCS HA solution

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

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
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

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)

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 an 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
```

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
```

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)
```

Glossary

UCA: Unified Correlation Analyzer

EBC: Event Based Correlation

VCS: Veritas Cluster Server

JDK: Java Development Kit

JRE: Java Runtime Environment

HA: High Availability