Using Oracle Clusterware to protect Oracle Application Server

An Oracle White Paper
November 2005

ORACLE

Using Oracle Clusterware to protect Oracle Application Server

EXECUTIVE OVERVIEW

Using the Oracle Clusterware 10*g* Release 2, together with Oracle Application Server 10*g* Release 2 offers customers a high availability solution fully developed and supported by Oracle.

- Customers can use OracleAS in an active-passive configuration in the same hardware resource as the Database, and protect the OracleAS components with Oracle Clusterware, saving money by not using hardware load balancer and also reducing the licensing cost of OracleAS. Alternatively, customers can opt for an active-active configuration for OracleAS and use a front-end software load balancer (Webcache) protected by Oracle Clusterware thus leveraging the scalability of an active-active solution without incurring in the considerable cost of redundant hardware load balancers.
- The complete application environment can be protected with a single product without needing to learn about, or incur the cost of, third party technologies . This reduces the total cost of ownership of the solution (lower administration costs, lower maintenance costs, lower installation costs)
- Easy to install and implement using the Oracle Universal Installer
- Proven technology - the High Availability API and Oracle Clusterware have been in use since Oracle Database 10g Release 1, where they are used to protect the nodeapps/instance/listener/VIP resources, and OracleCRS has been extensively tested with Oracle Real Application Clusters.
- Oracle provides the same Oracle Clusterware on all supported operating systems. Heterogeneous environments increase the deployment and maintenances costs of a topology. By using the same clusterware for the Oracle Database and Oracle OAS customers only need to learn once about the clusterware.
- We provide a VIP implementation that is able to fail over from network interface card to network interface card (redundancy) and from node to node. This is a unique selling proposition for the Oracle VIP implementation.

This white paper provides action scripts for two examples. These action scripts can easily modified to protect other or more components.

ORACLE CLUSTERWARE

With Oracle Clusterware 10*g* Release 2, Oracle Clusterware can be implemented independently of Real Application Clusters. Oracle Clusterware includes a high availability framework and API that allows any application process to be put under the control of the Oracle Clusterware. This provides very high availability for all your applications as the Oracle Clusterware will monitor the application and restart or relocate it in the case of a failure.

HIGH AVAILABILITY API

The High Availability API is an application-programming interface to allow processes to be put under the High Availability infrastructure that is part of the Oracle Clusterware distributed with Oracle Database 10g. A user written script defines how Oracle Clusterware should start, stop and relocate the process when the cluster node status changes. This extends the high availability services of the cluster to any application running on the cluster. Therefore the High Availability API can be used to increase the availability of Oracle Application Server.

To make applications highly available, first create an application profile that identifies your application. The application profile uses a second component, an action program, that describes how the Oracle Clusterware should monitor your application and how the Oracle Clusterware should respond to changes in your application's status. Oracle stores application profile attributes in the OCR. The definitions of an application profile, action program, and the other primary Oracle Clusterware high availability components are as follows:

- **Action Program** - A program that defines the location of your program, the monitoring frequency, and the start and stop actions that the Oracle Clusterware should perform on the application. The start action starts the application, the stop action stops the application, and the monitoring or check action checks the application's status.
- **Application Profile** - An Oracle Clusterware resource file that describes the attributes of your application. An application profile influences your application's behavior and it identifies other programs or scripts that the Oracle Clusterware should run to perform various actions.
- **Privileges** - Access and usage privileges that enable the Oracle Clusterware to control all of the components of your application for high availability operations, including the right to start processes under other user identities. The Oracle Clusterware must run as a privileged user to control applications with the correct start and

stop processes. On UNIX-based platforms, this usually implies that the Oracle Clusterware must run as the root user and on Windows-based platforms the Oracle Clusterware must run as LocalSystem.

- **Resource** - An entity that the Oracle Clusterware manages for high availability such as your application.
- **Resource Dependency** - A relationship among resources or applications that implies an operational ordering. For example, during a start operation, parent resources are started before resources that have dependencies. During a stop, the most dependent resources are stopped first.
- **Oracle Cluster Registry (OCR)** - A mechanism that stores configuration information that the Oracle Clusterware and other RAC manageability systems use. The OCR uses a hierarchical namespace for key value pairs. Keys and subkeys have enforced user, group, and other permissions. The OCR is stored either on a shared raw partition or on a non-cached, shared filesystem file.
- **Template** - A user-created text file that contains the default values for application profile attributes. Template files contain default values for profile attributes.

HOW TO CREATE A USRVIP OR APPSVIP AND APPLICATION PROFILE

Create a USRVIP or APPSVIP

If your application is accessed by way of a network, then Oracle recommends that you create a virtual internet protocol address for the application as a dependent resource. The advantage of using the Oracle VIP implementation is that we build our VIP implementation reliable in several cases. The Oracle VIP is able to fail over from network interface card to network interface card within a given node (NIC redundancy) and is able to fail over from node to node in case all public networks are down in a given node.

As operating system user oracle create application VIP addresses as follows:

1.
```
$ORA_CRS_HOME/bin/crs_profile -create appsvip \
        -t application \
        -a $ORA_CRS_HOME/bin/usrvip \
        -o oi=eth0,ov=138.2.237.23,on=255.255.254.0
```
2.
```
$ORA_CRS_HOME/bin/crs_register appsvip
```

On UNIX-based operating systems, the application VIP address script must run as the root user. As the root user, change the owner of the resource as follows:

```
$ORA_CRS_HOME/bin/crs_setperm appsvip -o root
```

As the root user, enable the oracle user to run this script:

```
$ORA_CRS_HOME/bin/crs_setperm appsvip -u user:oracle:r-x
```

As the oracle user, start the VIP address as follows:

```
crs_start appsvip
```

To check wether the appsvip is online and online on which node run:

```
crs_stat -t | grep appsvip

HA Resource        Target     State
-----------        ------     -----
appsvip            ONLINE     ONLINE on stnsp012
```

Create an APPLICATION PROFILE

Application profiles have attributes that define how the Oracle Clusterware starts, manages, and monitors applications. One attribute is the location of the action program that the Oracle Clusterware uses to manipulate the application. The Oracle Clusterware uses the action program to monitor or check the application status and to start and stop it. Oracle reads application profiles from files stored in specified locations and stores the information in the OCR. You use the Oracle Clusterware commands within profiles to designate resource dependencies and to determine what happens to an application or service when it loses access to a resource on which it depends.

The filenames of application profiles must be in the form resource_name.cap where resource_name is the name that you or the system assigns to an application and cap is the file suffix. The Oracle Clusterware commands in profiles refer to applications by name, such as resource_name, but not by the full filename.

Application Resource Profiles

Attributes are defined by name=value entries in profile files and these entries can be in any order in the file. The following are some of the primary attributes of an application profile:

- Resources that are required by an application which are defined by settings for the REQUIRED_RESOURCES parameter. The Oracle Clusterware relocates or stops an application if a required resource becomes unavailable. Required resources are defined for each node.
- Rules for choosing the node on which to start or restart an application are defined by settings for the PLACEMENT parameter. The application must be accessible by the nodes that you nominate for placement.

- A list of nodes to use in order of preference when the Oracle Clusterware starts or fails over an application which is defined by settings for the HOSTING_MEMBERS parameter. This list is used if the placement policy defined by the PLACEMENT parameter is favored or restricted.

Required and Optional Profile Attributes

Application profiles have optional and required profile attributes. Optional profile attributes may be left unspecified in the profile. Optional profile attributes that have default values are merged at registration time with the values that are stored in the template for that resource type and for the generic template. Default values are derived from the template.

Each resource type has a template file named TYPE_resource_type.cap that is stored in the template subdirectory under the crs directory of the Oracle Clusterware home. A generic template file for values that are used in all types of resources is stored in the same location in the file named TYPE_generic.cap.

Default Profile Locations

Profiles may be located anywhere and need not be on a cluster-visible file system. RAC provides default locations for profiles as described in the next sub-section. The default location for profiles with root privileges on UNIX-based systems or Administrator privileges on Windows-based systems is the profile subdirectory under the crs directory of the Oracle Clusterware home. The default location for profiles with non-root or non-Administrator privileges is the public subdirectory under the crs directory of the Oracle Clusterware home.

Case 1 – Cold Failover Cluster active / passive example

Customers are using an active/passive solution and need to switch the Oracle Application Server processes to the new active node after the old active goes down. The install type for the below example was build on a Real Application Cluster Database hence there is no requirement to protect the Oracle Database, this is covered internally via Oracle Clusterware and CRS resources.

Download the oas_cluster.scr and oas_cluster_action.scr from the APPENDIX A to $ORA_CRS_HOME/crs/public and chmod +x. The script should be owned by Oracle. Distribute both scripts to all nodes. Modify path names if necessary.

1. create the profile

```
./crs_profile
-create oas_cluster
-t application
-r appsvip
-a /scratch/oracle-10.2.0/crs/crs/public/oas_cluster.scr
-o ci=5,ra=60
```

2. register the resource

```
crs_register oas_cluster
```

3. start the resource

```
crs_start oas_cluster
```

4. check if the resource is online

```
crs_stat –t | grep oas_cluster
```

```
HA Resource           Target     State
-----------           ------     -----
oas_cluster           ONLINE     ONLINE on stnsp012
```

To verify if the Oracle Application Server processes are running and the components are started use

/scratch/ias-10.2.0.2/opmn/bin/opmnctl status

```
Processes in Instance: J2EE.stnsp012.us.oracle.com
-------------------+----------------+---------+-----
ias-component      | process-type   |     pid | status
-------------------+----------------+---------+---------
DSA                | DSA            |     N/A | Down
LogLoader          | logloaderd     |     N/A | Down
dcm-daemon         | dcm-daemon     |    1971 | Alive
OC4J               | home           |     817 | Alive
WebCache           | WebCache       |     828 | Alive
WebCache           | WebCacheAdmin  |     818 | Alive
HTTP_Server        | HTTP_Server    |     820 | Alive
```

The action script oas_cluster_action.scr (APPENDIX A) is starting the iasconsole as well. To verify whether this is running use the below command.

/scratch/ias-10.2.0.2/bin/emctl status iasconsole Oracle Enterprise Manager 10g Application Server Control Release 10.1.2.0.2 Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://stnsp012.us.oracle.com:1156/emd/console/aboutApplication

Oracle Enterprise Manager 10g Application Server Control is running.

Logs are generated in directory /scratch/ias-10.2.0.2/sysman/log

Case 2 - Use SW loadbalancer instead of HW loadbalancer

Oracle Application Server Release 2 provides Oracle Application Server Web Cache for content caching. Besides acting as a cache, Web Cache can work as a compressor for the delivered content and as a load balancer itself. Web Cache is typically deployed on a set of machines separate from the rest of the Middle Tier.

In previous releases, you could configure OracleAS Web Cache solely as a software load balancer or reverse proxy in place of hardware load balancers. By applying a patch to this release, you can now configure OracleAS Web Cache as software load balancer or reverse proxy even in front of an application using Edge Side Includes (ESI) or in front of another OracleAS Web Cache forming a cache hierarchy.

http://download-uk.oracle.com/docs/cd/B15790_09/relnotes.1012/relnotes/wcache.htm#i1011446

The install type for the below example was build on a Real Application Cluster Database hence there is no requirement to protect the Oracle Database this is covered internally via Oracle Clusterware and CRS resources.

Download the webcache.scr and webcache_action.scr from the APPENDIX B to $ORA_CRS_HOME/crs/public and chmod +x. The script should be owned by Oracle. Distribute both scripts to all nodes. Modify path names if necessary.

1. create the profile
```
./crs_profile
-create webcache
-t application
-r appsvip
-a /scratch/oracle-10.2.0/crs/crs/public/webcache.scr
-o ci=5,ra=60
```
2. register the resource
```
crs_register webcache
```
3. start the resource
```
crs_start webcache
```
4. check if the resource is online
```
crs_stat -t | grep webcache

HA Resource           Target     State
-----------           ------     -----
webcache              ONLINE     ONLINE on stnsp012
```

To verify if the Oracle Application Server processes are running and the components are started use

/scratch/ias-10.2.0.2/opmn/bin/opmnctl status

```
Processes in Instance: J2EE.stnsp012.us.oracle.com
-------------------+----------------+---------+-----
ias-component      | process-type   |     pid | status
-------------------+----------------+---------+---------
WebCache           | WebCache       |   19693 | Alive
WebCache           | WebCacheAdmin  |   19686 | Alive
HTTP_Server        | HTTP_Server    |     N/A | Down
```

CONCLUSION

With Oracle Clusterware 10*g* Release 2 and Oracle Application Server customers are able to build a high availability infrastructure with a software stack from a single vendor, Oracle.

APPENDIX A

oas_cluster_action.scr (example)

```
#!/bin/sh
# ****************************************************************
# *                                                              *
# *    Copyright (c) 2002, 2003 Oracle Corporation.              *
# *         All rights reserved.                                 *
# *                                                              *
# *    Copyright (c) 1991, 1999, 2002 Digital Equipment          *
# *                   Corporation                                *
# *                                                              *
# *                                                              *
# *   All Rights Reserved.  Unpublished rights  reserved  under  *
# *   the copyright laws of the United States.                   *
# *                                                              *
# *   The software contained on this media  is  proprietary  to  *
# *   and  embodies  the  confidential  technology  of  Digital  *
# *   Equipment Corporation and Oracle Corporation.  Possession,  *
# *   use, duplication or dissemination of the software          *
# *   and media is authorized only pursuant to a valid written   *
# *   license from Digital Equipment Corporation and Oracle      *
# *   Corporation                                                *
# *                                                              *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure  *
# *   by the U.S. Government is subject to restrictions  as  set  *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013,  *
# *   or  in  FAR 52.227-19, as applicable.                      *
# *                                                              *
# *                                                              *
# ****************************************************************
#

SCRIPT=$0
ACTION=$1                                   # Action (start, stop or check)

ORA_OWNER=oracle10                          # ORACLE installation owner
ORA_APPL=/scratch/ias-10.2.0.2              # ORACLE_HOME of application

RET1=1                              # Internal return values ( do not change )
RET2=1                              # Internal return values ( do not change )
RETVAL=1                            # Script return value

#############################################################################
#
# Main section of Action Script - starts, stops, or checks an application
#
# This script is invoked by CRS when managing the application associated
# with this script.
#
# Argument:  $1 - start | stop | check
#
# Returns:   0 - successful start, stop, or check
#            1 - error
#
#############################################################################

#
# Start section - start the process and report results
#
case $1 in
'start')
       ulimit -n 65536
       ulimit -u unlimited

         echo "DATE: `date`" >> /tmp/e
         # A) START - OAS:
         $ORA_APPL/opmn/bin/opmnctl startall        1>/dev/null 2>&1
         RET1=$?

         echo "START-OAS: $RET1" >> /tmp/e

         # B) START - OASCONSOLE:
          $ORA_APPL/bin/emctl start iasconsole      1>/dev/null 2>&1
         RET2=$?

         echo "START-C: $RET2" >> /tmp/e

         # Prepare return values:
         if [ ${RET1:-0} -eq 0 ] && [ ${RET2:-0} -eq 0 ]; then
```

```
           RETVAL=0
         else
           RETVAL=1
         fi
    ;;

#
# Stop section - stop the process and report results
#
'stop')
         # A) STOP - OAS:
         $ORA_APPL/opmn/bin/opmnctl stopall         1>/dev/null 2>&1
         RET1=$?

         # B) STOP - OASCONSOLE:
         $ORA_APPL/bin/emctl stop iasconsole        1>/dev/null 2>&1
         RET2=$?

         # Prepare return values:
        if [ ${RET1:-0} -eq 0 ] && [ ${RET2:-0} -eq 0 ]; then
           RETVAL=0
        else
           RETVAL=1
         fi
    ;;

*)
    echo "usage: $0 {start stop}"
    ;;

esac
echo "RETURN: $RETVAL" >> /tmp/e
#
# Return value to CRS daemon:
#
echo "RETVAL: $RETVAL" >> /tmp/e
if [ $RETVAL -eq 0 ]; then
   exit 0
else
   exit 1
fi
#exit 0
```

oas_cluster.scr (example)

```
#!/bin/ksh -p
# ***************************************************************
# *                                                             *
# *    Copyright (c) 2002, 2003 Oracle Corporation.             *
# *         All rights reserved.                                *
# *                                                             *
# *    Copyright (c) 1991, 1999, 2002 Digital Equipment         *
# *                Corporation                                  *
# *                                                             *
# *                                                             *
# *   All Rights Reserved.  Unpublished rights  reserved  under *
# *   the copyright laws of the United States.                  *
# *                                                             *
# *   The software contained on this media  is  proprietary  to *
# *   and  embodies  the  confidential  technology  of  Digital *
# *   Equipment Corporation and Oracle Corporation.  Possession, *
# *   use, duplication or dissemination of the software         *
# *   and media is authorized only pursuant to a valid written  *
# *   license from Digital Equipment Corporation and Oracle     *
# *   Corporation                                               *
# *                                                             *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure *
# *   by the U.S. Government is subject to restrictions  as  set *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013, *
# *   or  in  FAR 52.227-19, as applicable.                     *
# *                                                             *
# *                                                             *
# ***************************************************************
#
# @(#)$RCSfile: crstmpl.scr $ $Revision: has/crs/template/crstmpl.scr#0 $ (DEC)
$Date: 21-feb-2005.14:52:54 $
#
#
#
######################################################################
#
# The following section contains variables that can be set to best
# suit your application.
#
# Please review each variable and set as needed.
#
# Set CAA_SCRIPT_DEBUG when invoking the script from command line
# for testing. This will cause all output events to go to the terminal,
# rather than being sent to EVM.
#
######################################################################
#
# Application name - set this variable to a name that describes this
# (mandatory)        application.  Enclose the name in double quotes.
#                    Examples: "apache", "netscape"

SERVICE_NAME="oascluster"

# Associated Processes - the application configured may consist of
# (mandatory         single or multiple processes.  Specifying the names
#                    of the processes here allows CAA to monitor that they
#                    are running and allows CAA to completely clean up when
#                    stopping the application.
#                    Ex:  "proc1 proc2"

PROBE_PROCS="opmn"

# Application Startup Command - CAA will invoke this command when starting
# (mandatory)        the application.  Include the command to execute along
#                    with any flags and arguments needed.  Use this
#                    variable along with START_APPCMD2 (see below) when
#                    dealing with a simple application start procedure.
#
#                    If the start procedure is complicated and/or involves
#                    many commands, you may find it easier to disregard
#                    this variable and manually code the commands needed
#                    in the "Start" section of this script (see below).
#
#                    Another alternative for a complicated start procedure
#                    is to create a separate script containing those
#                    commands and specifying that script in this variable.
#
#                    Ex: "/cludemo/avs/avsetup -s"
#
#                    NOTE: if not set, you must manually code the commands
```

```
#                   to start the application in the "Start" section of
#                   this script.

START_APPCMD="/scratch/oracle-10.2.0/crs/crs/public/oas_cluster_action.scr start"

# Secondary Application Startup Command - used in conjunction with the
# (optional)        Application Startup Command just described above.  Use
#                   if desired to implement a two-step startup process for
#                   your application, if needed.

START_APPCMD2=""

# Application Stop Command - CAA will invoke this command when stopping
# (optional)        the application.  Include the command to execute along
#                   with any flags and arguments needed.  Use this
#                   variable along with STOP_APPCMD2 (see below) when
#                   dealing with a simple application stop procedure.
#
#                   If the stop procedure is complicated and/or involves
#                   many commands, you may find it easier to disregard
#                   this variable and manually code the commands needed
#                   in the "Stop" section of this script (see below).
#
#                   Another alternative for a complicated stop procedure
#                   is to create a separate script containing those
#                   commands and specifying that script in this variable.
#
#                   Ex: "/cludemo/avs/avsetup -k"
#
#                   NOTE: if not set, you should manually code the commands
#                   to stop the application in the "Stop" section of
#                   this script.  Otherwise, this script will not stop the
#                   application in a graceful manner.

STOP_APPCMD="/scratch/oracle-10.2.0/crs/crs/public/oas_cluster_action.scr stop"

# Secondary Application Stop Command - used in conjunction with the
# (optional)        Application Stop Command just described above.  Use
#                   if desired to implement a two-step stop process for
#                   your application, if needed.

STOP_APPCMD2=""

# Application Directory - If set, this script will change to this directory
# (optional)        prior to executing the start process.  This may allow
#                   you to not have to specify full path names for
#                   commands or files in this directory.
#
#                   Ex:  "/var/opt/product1"

APPDIR=""

export SERVICE_NAME START_APPCMD START_APPCMD2
export APPDIR PROBE_PROCS STOP_APPCMD STOP_APPCMD2

UNAME="/bin/uname"
if [ ! -x "$UNAME" ]; then
 UNAME="/usr/bin/uname"
fi

if [ `$UNAME` = "HP-UX" ]; then
  export UNIX95=XPG4
fi

################################################################
#
# The following section contains variables used by CAA.  We recommend
# leaving them defined as is.
#
################################################################

DEBUG_PRIORITY=100
INFO_PRIORITY=200
ERROR_PRIORITY=500

SCRIPT=$0
ACTION=$1                    # Action (start, stop or check)

EVMPOST="evmpost"    # EVM command to post events
DEBUG=0

if [[ "$CAA_SCRIPT_DEBUG" != "" ]]; then
    DEBUG=1
```

```
    EVMPOST="evmpost -r | evmshow -D"
fi

export EVMPOST ACTION SCRIPT

#####################################################################
#
# The following section contains procedures that are available to
# be used from the start, stop, and check portions of this script.
#
#####################################################################

#
# postevent - Posts EVM event with specified parameters
#
# Argument:  $1 - priority (optional)
#            $2 - message  (optional)
#
#

postevent () {
    typeset pri=$1
    typeset msg=${2:-failed}

    typeset evt='event { name sys.ora.clu.crs.action_script '

    if [ ! -z "$pri" ]; then
        evt="$evt priority $pri "
    fi

    evt="$evt var {name name value \\\"$SERVICE_NAME\\\" } "
    evt="$evt var {name script value \\\"$SCRIPT\\\" } "
    evt="$evt var {name action value \\\"$ACTION\\\" } "
    evt="$evt var {name message value \\\"$msg\\\" }"

    evt="$evt }"

    evt="echo $evt | $EVMPOST"

    eval $evt
}

#
# getpid - list PIDs of all processes with supplied name
#
# Modified /sbin/init.d/bin/getpid to list all matches
#
# Arguments:  process name
#

getpid () {
    if [ -n "$1" ]; then
        GETMYPID=$1
        shift
        /bin/ps -e -o pid,comm | while read mypid command args
        do
            if [ "$command" = "$GETMYPID" ]; then
               echo "$mypid"
            fi
        done
    fi
}

#
# checkdaemon - return the number of instances of a daemon
#
# Argument:  process name
# Return:    number of instances of the named daemon currently running
#

checkdaemon () {
    R=`getpid $1 | wc -l`
    return $R
}

#
# zapdaemon - kill a given process using brutal force (i.e. -9)
#
# Argument:  list of processes to kill
# Return:    1 - failed to kill some process
#            0 - killed all processes
#
```

```
zapdaemon () {
    typeset ret=0

    for i in ${1}
    do
         checkdaemon ${i}
         if [ $? -ne 0 ]; then
             kill `getpid ${i}`
              checkdaemon ${i}
              if [ $? -ne 0 ]; then
                   kill -9 `getpid ${i}`
                   checkdaemon ${i}
                   if [ $? -ne 0 ]; then
                        postevent $ERROR_PRIORITY "${i}: stuck - could not kill -
KILL"
                        ret=1
                   else
                        postevent $ERROR_PRIORITY "${i}: killed with -KILL"
                   fi
              else
                  postevent "" "${i}: killed"
              fi
         fi
    done
    return $ret
}


#
# probeapp - Probe process to see if in process list.
#
# This simple form of process probing searches the process list for an
# entry corresponding to the specified process.  If found, all is assumed
# to be well.
#
# More accurate process probing may be available depending upon the nature
# of your application.  For instance, you might invoke a test command that
# the process should respond to and check the returned results.  You should
# consider adding this type of probing to this script, if possible.
#
# Argument:  process name
#
# Return:    1 - process not running
#            0 - process running
#

probeapp () {
    checkdaemon $1
    if [ $? -ne 0 ]; then
         postevent $DEBUG_PRIORITY "$1 check OK"
         return 0
    else
         postevent $DEBUG_PRIORITY "$1 check failed"
         return 1
    fi
}

##############################################################################
#
# Main section of Action Script - starts, stops, or checks an application
#
# This script is invoked by CAA when managing the application associated
# with this script.
#
# Argument:  $1 - start | stop | check
#
# Returns:   0 - successful start, stop, or check
#            1 - error
#
##############################################################################

#
# Start section - start the process and report results
#
# If the Application Startup Commands (see description above) were used,
# little, if any modifications are needed in this section.  If not used,
# you may replace most of the contents in this section with your own
# start procedure code.
#
# For improved serviceability, preserve the commands below that log
# messages or posts events.
#
```

```
case $1 in
'start')
    postevent $DEBUG_PRIORITY "trying to start"
    cd $APPDIR
    if [ "$START_APPCMD" != "" ]; then
         out=`$START_APPCMD`
         if [ $? -ne 0 ]; then
             postevent $ERROR_PRIORITY "start: $out"
             exit 1
         fi
    fi

    if [ "$START_APPCMD2" != "" ]; then
         out=`$START_APPCMD2`
         if [ $? -ne 0 ]; then
             postevent $ERROR_PRIORITY "start 2: $out"
             exit 1
         fi
    fi
    ;;

#
# Stop section - stop the process and report results
#
# If the Application Stop Commands or Associated Processes (see descriptions
# above) were used,little, if any modifications are needed in this section.
# If not used, you may replace most of the contents in this section with
# your own stop procedure code.
#
# For improved serviceability, preserve the commands below that log
# messages or posts events.
#

'stop')
    postevent $DEBUG_PRIORITY "trying to stop"
    cd $APPDIR
    if [ "$STOP_APPCMD" != "" ]; then
         out=`$STOP_APPCMD`
         if [ $? -ne 0 ]; then
             postevent $ERROR_PRIORITY "stop: $out"
             exit 1
         fi
    fi

    if [ "$STOP_APPCMD2" != "" ]; then
         out=`$STOP_APPCMD2`
         if [ $? -ne 0 ]; then
             postevent $ERROR_PRIORITY "stop 2: $out"
             exit 1
         fi
    fi

#
# Kill stubborn processes and applications that don't have a stop command
#

    for i in ${PROBE_PROCS}; do
         zapdaemon ${i}
    done

    ;;

#
# Check section - check the process and report results
#
# If you specified $PROBE_PROCS (see earlier description), very little,
# if any, changes are needed to have simple process checking.
#
# Your application might allow you to implement more accurate process
# checking.  If so, you may choose to implement that code here.  See the
# description for the probeapp function earlier in this script.
#
'check')
    for i in ${PROBE_PROCS}; do
         postevent $DEBUG_PRIORITY "trying to check $i"
         probeapp $i
         if [ $? -ne 0 ]; then
             postevent "" "check failed for $i"
             exit 1
         fi
    done

    ;;
```

```
*)
    postevent $ERROR_PRIORITY "usage: $0 {start stop check}"
    exit 1

    ;;
esac

postevent "" success
exit 0
```

APPENDIX B

webcache_action.scr (example)

```
#!/bin/sh
# ****************************************************************
# *                                                              *
# *    Copyright (c) 2002, 2003 Oracle Corporation.              *
# *         All rights reserved.                                 *
# *                                                              *
# *    Copyright (c) 1991, 1999, 2002 Digital Equipment          *
# *                   Corporation                                *
# *                                                              *
# *                                                              *
# *   All Rights Reserved.  Unpublished rights  reserved  under  *
# *   the copyright laws of the United States.                   *
# *                                                              *
# *   The software contained on this media  is  proprietary  to  *
# *   and  embodies  the  confidential  technology  of  Digital  *
# *   Equipment Corporation and Oracle Corporation.  Possession,  *
# *   use, duplication or dissemination of the software          *
# *   and media is authorized only pursuant to a valid written   *
# *   license from Digital Equipment Corporation and Oracle      *
# *   Corporation                                                *
# *                                                              *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure  *
# *   by the U.S. Government is subject to restrictions  as  set  *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013,  *
# *   or  in  FAR 52.227-19, as applicable.                      *
# *                                                              *
# *                                                              *
# ****************************************************************
#

SCRIPT=$0
ACTION=$1                                                       # Action
(start, stop or check)

ORA_OWNER=oracle10                                              # ORACLE
installation owner
ORA_APPL=/scratch/ias-10.2.0.2                                  #
ORACLE_HOME of application

RET1=1                                                          #
Internal return values ( do not change )
RETVAL=1                                                        # Script
return value

#############################################################################
#
# Main section of Action Script - starts, stops, or checks an application
#
# This script is invoked by CRS when managing the application associated
# with this script.
#
# Argument:  $1 - start | stop | check
#
# Returns:   0 - successful start, stop, or check
#            1 - error
#
#############################################################################

#
# Start section - start the process and report results
#
case $1 in
'start')
       ulimit -n 65536
       ulimit -u unlimited

        echo "DATE: `date`" >> /tmp/e
        echo "ulimit: `ulimit -n`" >> /tmp/e
        echo "ulimit: `ulimit -u`" >> /tmp/e

        # A) START - WEBCACHE:
        $ORA_APPL/opmn/bin/opmnctl startproc ias-component=WebCache
1>/dev/null 2>&1
        RET1=$?


        # Prepare return values:
```

```
        if [ ${RET1:-0} -eq 0 ];  then
           RETVAL=0
        else
           RETVAL=1
        fi
    ;;

#
# Stop section - stop the process and report results
#
'stop')
        # A) STOP - WEBCACHE:
        $ORA_APPL/opmn/bin/opmnctl stopproc ias-component=WebCache    1>/dev/null
2>&1
        RET1=$?


        # Prepare return values:
        if [ ${RET1:-0} -eq 0 ]; then
           RETVAL=0
        else
           RETVAL=1
        fi
    ;;

*)
    echo "usage: $0 {start stop}"
    ;;

esac
echo "RETURN: $RETVAL" >> /tmp/e
#
# Return value to CRS daemon:
#
echo "RETVAL: $RETVAL" >> /tmp/e
if [ $RETVAL -eq 0 ]; then
   exit 0
else
   exit 1
fi
#exit 0
```

webcache.scr (example)

```
#!/bin/ksh -p
# ************************************************************************
# *                                                                      *
# *    Copyright (c) 2002, 2003 Oracle Corporation.                      *
# *         All rights reserved.                                         *
# *                                                                      *
# *    Copyright (c) 1991, 1999, 2002 Digital Equipment                  *
# *                   Corporation                                        *
# *                                                                      *
# *                                                                      *
# *   All Rights Reserved.  Unpublished rights  reserved  under          *
# *   the copyright laws of the United States.                           *
# *                                                                      *
# *   The software contained on this media  is  proprietary  to          *
# *   and  embodies  the  confidential  technology  of  Digital          *
# *   Equipment Corporation and Oracle Corporation.  Possession,         *
# *   use, duplication or dissemination of the software                  *
# *   and media is authorized only pursuant to a valid written           *
# *   license from Digital Equipment Corporation and Oracle              *
# *   Corporation                                                        *
# *                                                                      *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure         *
# *   by the U.S. Government is subject to restrictions  as  set         *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013,         *
# *   or  in  FAR 52.227-19, as applicable.                              *
# *                                                                      *
# *                                                                      *
# ************************************************************************
#
# @(#)$RCSfile: crstmpl.scr $ $Revision: has/crs/template/crstmpl.scr#0 $ (DEC)
$Date: 21-feb-2005.14:52:54 $
#
#
#
######################################################################
#
# The following section contains variables that can be set to best
# suit your application.
#
# Please review each variable and set as needed.
#
# Set CAA_SCRIPT_DEBUG when invoking the script from command line
# for testing. This will cause all output events to go to the terminal,
# rather than being sent to EVM.
#
######################################################################
#
# Application name - set this variable to a name that describes this
# (mandatory)        application.  Enclose the name in double quotes.
#                    Examples: "apache", "netscape"

SERVICE_NAME="webcache"

# Associated Processes - the application configured may consist of
# (mandatory         single or multiple processes.  Specifying the names
#                    of the processes here allows CAA to monitor that they
#                    are running and allows CAA to completely clean up when
#                    stopping the application.
#                    Ex:  "proc1 proc2"

PROBE_PROCS="webcached"

# Application Startup Command - CAA will invoke this command when starting
# (mandatory)        the application.  Include the command to execute along
#                    with any flags and arguments needed.  Use this
#                    variable along with START_APPCMD2 (see below) when
#                    dealing with a simple application start procedure.
#
#                    If the start procedure is complicated and/or involves
#                    many commands, you may find it easier to disregard
#                    this variable and manually code the commands needed
#                    in the "Start" section of this script (see below).
#
#                    Another alternative for a complicated start procedure
#                    is to create a separate script containing those
#                    commands and specifying that script in this variable.
#
#                    Ex: "/cludemo/avs/avsetup -s"
#
#                    NOTE: if not set, you must manually code the commands
#                    to start the application in the "Start" section of
```

```
#                   this script.

START_APPCMD="/scratch/oracle-10.2.0/crs/crs/public/webcache_action.scr start"

# Secondary Application Startup Command - used in conjunction with the
# (optional)         Application Startup Command just described above.  Use
#                    if desired to implement a two-step startup process for
#                    your application, if needed.

START_APPCMD2=""

# Application Stop Command - CAA will invoke this command when stopping
# (optional)         the application.  Include the command to execute along
#                    with any flags and arguments needed.  Use this
#                    variable along with STOP_APPCMD2 (see below) when
#                    dealing with a simple application stop procedure.
#
#                    If the stop procedure is complicated and/or involves
#                    many commands, you may find it easier to disregard
#                    this variable and manually code the commands needed
#                    in the "Stop" section of this script (see below).
#
#                    Another alternative for a complicated stop procedure
#                    is to create a separate script containing those
#                    commands and specifying that script in this variable.
#
#                    Ex: "/cludemo/avs/avsetup -k"
#
#                    NOTE: if not set, you should manually code the commands
#                    to stop the application in the "Stop" section of
#                    this script.  Otherwise, this script will not stop the
#                    application in a graceful manner.

STOP_APPCMD="/scratch/oracle-10.2.0/crs/crs/public/webcache_action.scr stop"

# Secondary Application Stop Command - used in conjunction with the
# (optional)         Application Stop Command just described above.  Use
#                    if desired to implement a two-step stop process for
#                    your application, if needed.

STOP_APPCMD2=""

# Application Directory - If set, this script will change to this directory
# (optional)         prior to executing the start process.  This may allow
#                    you to not have to specify full path names for
#                    commands or files in this directory.
#
#                    Ex:  "/var/opt/product1"

APPDIR=""

export SERVICE_NAME START_APPCMD START_APPCMD2
export APPDIR PROBE_PROCS STOP_APPCMD STOP_APPCMD2

UNAME="/bin/uname"
if [ ! -x "$UNAME" ]; then
 UNAME="/usr/bin/uname"
fi

if [ `$UNAME` = "HP-UX" ]; then
  export UNIX95=XPG4
fi

################################################################
#
# The following section contains variables used by CAA.  We recommend
# leaving them defined as is.
#
################################################################

DEBUG_PRIORITY=100
INFO_PRIORITY=200
ERROR_PRIORITY=500

SCRIPT=$0
ACTION=$1                    # Action (start, stop or check)

EVMPOST="evmpost"    # EVM command to post events
DEBUG=0

if [[ "$CAA_SCRIPT_DEBUG" != "" ]]; then
    DEBUG=1
    EVMPOST="evmpost -r | evmshow -D"
```

```
fi

export EVMPOST ACTION SCRIPT

#####################################################################
#
# The following section contains procedures that are available to
# be used from the start, stop, and check portions of this script.
#
#####################################################################

#
# postevent - Posts EVM event with specified parameters
#
# Argument:  $1 - priority (optional)
#            $2 - message  (optional)
#
#

postevent () {
    typeset pri=$1
    typeset msg=${2:-failed}

    typeset evt='event { name sys.ora.clu.crs.action_script '

    if [ ! -z "$pri" ]; then
        evt="$evt priority $pri "
    fi

    evt="$evt var {name name value \\\"$SERVICE_NAME\\\" } "
    evt="$evt var {name script value \\\"$SCRIPT\\\" } "
    evt="$evt var {name action value \\\"$ACTION\\\" } "
    evt="$evt var {name message value \\\"$msg\\\" }"

    evt="$evt }"

    evt="echo $evt | $EVMPOST"

    eval $evt
}

#
# getpid - list PIDs of all processes with supplied name
#
# Modified /sbin/init.d/bin/getpid to list all matches
#
# Arguments:  process name
#

getpid () {
    if [ -n "$1" ]; then
        GETMYPID=$1
        shift
        /bin/ps -e -o pid,comm | while read mypid command args
        do
            if [ "$command" = "$GETMYPID" ]; then
                echo "$mypid"
            fi
        done
    fi
}

#
# checkdaemon - return the number of instances of a daemon
#
# Argument:  process name
# Return:    number of instances of the named daemon currently running
#

checkdaemon () {
    R=`getpid $1 | wc -l`
    return $R
}

#
# zapdaemon - kill a given process using brutal force (i.e. -9)
#
# Argument:  list of processes to kill
# Return:    1 - failed to kill some process
#            0 - killed all processes
#

zapdaemon () {
```

```
    typeset ret=0

    for i in ${1}
    do
        checkdaemon ${i}
        if [ $? -ne 0 ]; then
            kill `getpid ${i}`
            checkdaemon ${i}
            if [ $? -ne 0 ]; then
                kill -9 `getpid ${i}`
                checkdaemon ${i}
                if [ $? -ne 0 ]; then
                    postevent $ERROR_PRIORITY "${i}: stuck - could not kill -
KILL"
                    ret=1
                else
                    postevent $ERROR_PRIORITY "${i}: killed with -KILL"
                fi
            else
                postevent "" "${i}: killed"
            fi
        fi
    done
    return $ret
}


#
# probeapp - Probe process to see if in process list.
#
# This simple form of process probing searches the process list for an
# entry corresponding to the specified process.  If found, all is assumed
# to be well.
#
# More accurate process probing may be available depending upon the nature
# of your application.  For instance, you might invoke a test command that
# the process should respond to and check the returned results.  You should
# consider adding this type of probing to this script, if possible.
#
# Argument:  process name
#
# Return:    1 - process not running
#            0 - process running
#

probeapp () {
    checkdaemon $1
    if [ $? -ne 0 ]; then
        postevent $DEBUG_PRIORITY "$1 check OK"
        return 0
    else
        postevent $DEBUG_PRIORITY "$1 check failed"
        return 1
    fi
}

###############################################################################
#
# Main section of Action Script - starts, stops, or checks an application
#
# This script is invoked by CAA when managing the application associated
# with this script.
#
# Argument:  $1 - start | stop | check
#
# Returns:   0 - successful start, stop, or check
#            1 - error
#
###############################################################################

#
# Start section - start the process and report results
#
# If the Application Startup Commands (see description above) were used,
# little, if any modifications are needed in this section.  If not used,
# you may replace most of the contents in this section with your own
# start procedure code.
#
# For improved serviceability, preserve the commands below that log
# messages or posts events.
#

case $1 in
```

```
'start')
    postevent $DEBUG_PRIORITY "trying to start"
    cd $APPDIR
    if [ "$START_APPCMD" != "" ]; then
        out=`$START_APPCMD`
        if [ $? -ne 0 ]; then
            postevent $ERROR_PRIORITY "start: $out"
            exit 1
        fi
    fi

    if [ "$START_APPCMD2" != "" ]; then
        out=`$START_APPCMD2`
        if [ $? -ne 0 ]; then
            postevent $ERROR_PRIORITY "start 2: $out"
            exit 1
        fi
    fi
    ;;

#
# Stop section - stop the process and report results
#
# If the Application Stop Commands or Associated Processes (see descriptions
# above) were used,little, if any modifications are needed in this section.
# If not used, you may replace most of the contents in this section with
# your own stop procedure code.
#
# For improved serviceability, preserve the commands below that log
# messages or posts events.
#

'stop')
    postevent $DEBUG_PRIORITY "trying to stop"
    cd $APPDIR
    if [ "$STOP_APPCMD" != "" ]; then
        out=`$STOP_APPCMD`
        if [ $? -ne 0 ]; then
            postevent $ERROR_PRIORITY "stop: $out"
            exit 1
        fi
    fi

    if [ "$STOP_APPCMD2" != "" ]; then
        out=`$STOP_APPCMD2`
        if [ $? -ne 0 ]; then
            postevent $ERROR_PRIORITY "stop 2: $out"
            exit 1
        fi
    fi

#
# Kill stubborn processes and applications that don't have a stop command
#

    for i in ${PROBE_PROCS}; do
        zapdaemon ${i}
    done

    ;;

#
# Check section - check the process and report results
#
# If you specified $PROBE_PROCS (see earlier description), very little,
# if any, changes are needed to have simple process checking.
#
# Your application might allow you to implement more accurate process
# checking.  If so, you may choose to implement that code here.  See the
# description for the probeapp function earlier in this script.
#
'check')
    for i in ${PROBE_PROCS}; do
        postevent $DEBUG_PRIORITY "trying to check $i"
        probeapp $i
        if [ $? -ne 0 ]; then
            postevent "" "check failed for $i"
            exit 1
        fi
    done

    ;;
```

```
*)
    postevent $ERROR_PRIORITY "usage: $0 {start stop check}"
    exit 1

    ;;
esac

postevent "" success
exit 0
```

REFERENCES

Oracle® Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide 10*g* Release 2 (10.2)

Using Oracle Clusterware to Protect 3rd Party Applications
http://www.oracle.com/technology/products/database/clustering/pdf/TWP_OracleClusterware3rdParty.pdf

Oracle Application Server 10g Release 2 High Availability
http://www.oracle.com/technology/products/ias/hi_av/OracleApplicationserver10gr2HA-WP.pdf

Using Oracle Application Server Web Cache as a Highly Available Load Balancer
http://www.oracle.com/technology/products/ias/hi_av/WC904-LBR-CFC.pdf

White Paper Title
November 2005
Author: Roland Knapp
Contributing Authors: Thomas Robert, Fermin Castro, Phil Newlan

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com