ORACLE

ORAchk User Guide

Oracle Configuration Audit Tool

Version 12.1.0.2.3

Revision 2

Contents

Purpose

This document provides users of the ORAchk (Oracle Configuration Audit Tool) the information needed in order to run and maintain the tool. The tool is designed to audit various important configuration settings within an Oracle System.

> **Note:** As of version 2.2.4 RACcheck has been renamed to ORAchk in order to reflect its expanded role for checking the health of a growing list of Oracle products. Please refer to **RACcheck Configuration Audit Tool Statement of Direction - name change to ORAchk (Doc ID 1591208.1)** for additional details. Version 12.1.0.2.3 will be the LAST release that the raccheck script will be distributed with ORAchk. Customers should transition over to use the orachk script in any automation or environmental configurations.

The tool audits configuration settings within the following categories:

1. OS kernel parameters
2. OS packages
3. Many other OS configuration settings
4. CRS/Grid Infrastructure
5. RDBMS
6. ASM
7. Database Parameters and Configuration Settings
8. Upgrade Readiness assessment for target versions 11.2.0.3 and above

Scope

The scope of the ORAchk health assessment tool is Oracle database servers, Grid Infrastructure, Oracle databases, hardware, Operating System and RAC software. Starting with ORAchk 2.2.0, ORAchk functionality was extended to Oracle Single Instance Databases, Oracle Restart Systems as well as RAC One node configurations.

Supported Platforms

At this time, the tool is supported on the following UNIX platforms:

- Intel Linux* (Enterprise Linux/RedHat 5, 6, 7 and SuSE 9,10, 11, 12)
- Linux on System Z (RedHat 6, 7 and SuSE 12)
- Oracle Solaris SPARC (Solaris 10 and 11)
- Oracle Solaris x86-64 (Solaris 10 and 11)
- AIX **
- HPUX**
- MS Windows (2008 and 2012)***

* No planned support for Linux Itanium, 32-bit Linux is NOT supported
** Requires BASH Shell 3.2 or higher to be installed on the systems
*** Requires Cygwin (See Appendix O for details)

Supported Database Versions

At this time, the tool is supported on the following database versions:

- 10gR2
- 11gR1
- 11gR2
- 12cR1

Components

The tool consists of a script and 2 driver files (as well as some other miscellaneous files).

1. orachk (and raccheck for backward compatibility)
2. collections.dat
3. rules.dat

Features

1. ORAchk is NON-INTRUSIVE and does not change anything in the environment, except as detailed below:
   - SSH user equivalence for the RDBMS software owner is assumed to be configured among all the database servers being audited in order for it to execute commands on the remote database server nodes. If the tool determines that this user equivalence is not established it will offer to set it up either temporarily or permanently at the option of the user. If the user chooses to set up SSH user equivalence temporarily then the script will do so for the duration of the execution of the tool but then it will return the system to the state in which it found SSH user equivalence originally. For those wishing to configure SSH user equivalence outside the tool (if not already configured), consult My Oracle Support Note: 372795.1.

     Note: SSH user equivalence is ONLY necessary for those systems running Grid Infrastructure for supporting a cluster, this is NOT required for Single Instance Databases or Oracle Restart Configurations.
   - ORAchk creates a number of small output files into which the data necessary to perform the assessment is collected
   - ORAchk creates and executes some scripts dynamically in order to accomplish some of the data collection
   - ORAchk cleans up after itself any temporary files that are created and not needed as part of the collection.

2. ORAchk interrogates the system to determine the status of the Oracle stack components (i.e., Grid Infrastructure, RDBMS, RAC, etc) and whether they are installed and/or running. Depending upon the status of each component, the tool runs the appropriate collections and audit checks. If due to local environmental configuration the tool is unable to properly determine the needed environmental information please refer to the TROUBLESHOOTING section.

3. Watchdog daemon - ORAchk automatically runs a daemon in the background to monitor command execution progress. If, for any reason, one of the commands run by the tool should hang or take longer than anticipated, the monitor daemon kills the hung command after a configurable timeout so that main tool execution can progress. If that happens then the collection or command that was hung is skipped and a notation is made in the final output report. If the default timeout is too short please see the TROUBLESHOOTING section regarding adjustment of the RAT_TIMEOUT, and RAT_ROOT_TIMEOUT parameters.

4. If ORAchk's driver files are older than 90 days, the driver files are considered to be "stale" and the script will notify the user of a stale driver file. A new version of the tool and its driver files (kit) must be obtained from My Oracle Support Note: 1268927.1.

5. When the ORAchk completes the collection and analysis it produces a detailed HTML formatted report. A output .zip file is also produced by ORAchk. This output .zip file can be provided to Oracle Support for further analysis if an SR needs to be logged. The detailed report will contain Benefit/Impact, Risk and Action/Repair information. In many cases it will also reference publicly available documents with additional information about the problem and how to resolve it.

6. The results of the audit checks can be optionally uploaded into database tables for reporting purposes. See below for more details on this subject.

7. In some cases customers may want to stage ORAchk on a shared filesystem so that it can be accessed from various systems but be maintained in a single location rather than being copied to each cluster on which it may be used. The default behavior of the tool is to create a subdirectory and its output files in the location where the tool is staged. If that staging area is a read only filesystem or if the user for any reason would like the output to be created elsewhere then there is an environment variable which can be used for that purpose. The RAT_OUTPUT parameter can be set to any valid writable location and the output will be created there.

Usage Considerations

- The default temporary working directory location is the home directory of the userid that launched ORAchk. For example, if the "oracle" userid was used to launch ORAchk and the home directory of the "oracle" userid is "/home/oracle", then the default temporary working directory location is "/home/oracle". It is possible to change the default location by setting the "RAT_TMPDIR" environment variable (e.g. export RAT_TMPDIR=/tmp) prior to executing ORAchk.

  Note: If you are using SUDO access for root (see Appendix M) and you wish to change the default temporary working directory location, you must also reflect your alternate location in the "/etc/sudoers" file. In this configuration your "/etc/sudoers" file on each server must contain the following line:

  ```
  oracle ALL=(root) NOPASSWD:/tmp/root_exachk.sh
  ```

- It is recommended that the kit be staged and operated from a local filesystem on a single database server in order to provide the best performance possible.

- For maximum usefulness execute ORAchk when the Grid Infrastructure and at least one database are up and running.

- While ORAchk is a minimal impact tool, it is a best practice to execute it during times of least load on the system.

- To avoid possible problems running the tool from terminal sessions on a network attached workstation or laptop, consider running the tool using VNC so that if there is a network interruption the tool will continue to process to completion.

- If the execution of the tool should fail for some reason it can be re-run from the beginning but the tool is not "resumable" from the point of failure.

- As of Release 2.2.1, ORAchk has the ability to execute on all nodes in parallel. In order to take advantage of the ROOT specific checks while still allowing parallel execution, you have the EXPECT utility installed on the system or configure SUDO for use by ORAchk as described in Appendix M.

- Starting with ORAchk version 2.2.5, database collections are now executed in parallel. To choose a non-default maximum number of slave processes (default is automatically calculated) for database collections use the –dbparallel [n]. For example to change the default calculated value to 30, you would execute "./orachk –dbparallel 30". If for some reason you would like to disable parallel database collections you would use the –dbserial option.

  Note: The higher the degree of parallelism, the more resources are consumed but the elapsed time is reduced. In addition to being able to raise the number of parallel slaves beyond the default value, you can also lower the number of parallel slaves below the default value. There are reasons to do one or the other. For example, after the entire system has been brought up after maintenance, but before actual users are permitted on the system, you might want to use a higher number of parallel slaves to finish an ORAchk run as quickly as possible. For another example, on a busy production system, you might use a number less than the default value, yet more than running in serial mode to get a run more quickly but with less of an impact on the running system.

- Beginning with 12.1.0.2.1, there are two basic execution models available for ORAchk

  - Execute as "root" userid - In this model, the "root" userid is used to launch a complete ORAchk run. The ORAchk process running under the "root" userid will then use the "su" command to execute commands as one of the lower privileged owners of the RDBMS or grid homes. The lower privileged accounts will not have the ability to elevate access in order to execute the "root" level checks. This approach can have advantages in role separated environments, or environments with more restrictive security models.

- Execute as RDBMS Home owner - In this method, ORAchk is launched from either the RDBMS or Grid home owner userid. The userid that launched ORAchk must have the ability to elevate access to “root” userid in order to run the “root” level checks. This can be accomplished by providing the “root” userid password, setting up “sudo” (as described in Appendix M), or passwordless SSH connectivity. This means that a lower privileged account must be configured to be able to switch to the “root” userid. This approach may require multiple runs in role separated environments, and more restrictive security requirements may not permit the elevation.

When to Run ORAchk

1. After initial Oracle RAC deployment
2. Before planned system maintenance
3. After planned system maintenance
4. At least once every three months

Usage

Stage and run the tool only on database servers as the Oracle RDBMS software owner (oracle) if Oracle software installed.

The tool can be run with following arguments:

```
Usage : ./orachk [-abvhpfmsuSo:c:t:]

        -h      Prints this page.
        -a      All (Perform best practice check and recommended patch check)
        -b      Best Practice check only. No recommended patch check
        -v      Show version
        -p      Patch check only
        -m      exclude checks for Maximum Availability Architecture (MAA) scorecards(see user guide for more details)
        -u      Run orachk to check pre-upgrade or post-upgrade best practices for 11.2.0.3 and above
                -o pre or -o post is mandatory with -u option like ./orachk -u -o pre
        -f      Run Offline.Checks will be performed on data already collected from the system
        -o      Argument to an option. if -o is followed by v,V,Verbose,VERBOSE or Verbose, it will print checks which passs on the
screen
                if -o option is not specified,it will print only failures on screen. for eg: orachk -a -o v

        -clusternodes
                Pass comma separated node names to run orachk only on subset of nodes.
        -dbnames
                Pass comma separated database names to run orachk only on subset of databases
        -localonly
                Run orachk only on local node.
        -debug
                Run orachk in debug mode. Debug log will be generated.
                eg:- ./orachk -debug
        -dbnone
                Do not prompt database selection and skip all database related checks.
        -dball
                Do not prompt database selection and run database related checks on all databases discovered on system.

        -c      Used only under the guidance of Oracle support or development to override default components

        -upgrade
                Used to force upgrade the version of orachk being run.
        -noupgrade
                Do not prompt for an upgrade even if a later version is available under the location specified by RAT_UPGRADE_LOC.


        Report Options:
        -nopass
                Skip PASS'ed check to print in orachk report and upload to database.

        -noscore
                Do not print healthscore in HTML report.

        -diff <Old Report> <New Report> [-outfile <Output HTML>]
                Diff two orachk reports. Pass directory name or zip file or html report file as <Old Report> & <New Report>

        -exadiff <Exalogic collection1> <Exalogic collection2>
                Compare two different Exalogic rack and see if both are from the same release.Pass directory name or zip file as
<Exalogic collection1> & <Exalogic collection2>(applicable for Exalogic only)
```

```
        -merge
                Pass comma separated collection names(directory or zip files) to merge collections and prepare single report.
                eg:- ./orachk -merge orachk_hostname1_db1_120213_163405.zip,orachk_hostname2_db2_120213_164826.zip

        -tag <tagname>
                Appends <tagname> to Report Name. <Tagname> must contain only alphanumeric characters.
                for eg: ./orachk -tag newtag123 will append 'newtag123' to report name like '<report name>_newtag123.html'
        Auto Restart Options:
                -auto_restart -h: Prints help for this option
                -<initsetup|initrmsetup|initcheck|initpresetup>
                initsetup       : Setup auto restart. Auto restart functionality automatically brings up orachk daemon when node starts
                initrmsetup     : Remove auto restart functionality
                initcheck       : Check if auto restart functionality is setup or not
                initpresetup    : Sets root user equivalency for COMPUTE, STORAGE and IBSWITCHES.(root equivalency for COMPUTE nodes is
mandatory for setting up auto restart functionality)
        Daemon Options:


        -d <start|start_debug|stop|status|info|stop_client|nextautorun|-h>
                start                   : Start the orachk daemon
                start_debug             : Start the orachk daemon in debug mode
                stop                    : Stop the orachk daemon
                status                  : Check if the orachk daemon is running
                info                    : Print information about running orachk daemon
                stop_client             : Stop the orachk daemon client

                nextautorun [-id <ID>]  : print the next auto run time
                                          if '-id <ID>' is specified, it will print the next auto run time for specified autorun
schedule ID

                -h                      : Prints help for this option


        -daemon
                run orachk only if daemon is running
        -nodaemon
                Dont use daemon to run orachk
        [-id <ID>] -set
                configure orachk daemon parameter like 'param1=value1;param2=value2... '
                if '-id <ID>' is specified, it will configure orachk daemon parameter(s) for specified autorun schedule ID
                 Supported parameters are:-

                 (Deprecated) - AUTORUN_INTERVAL <n[d|h]> :- Automatic rerun interval in daemon mode.Set it zero to disable automatic
rerun which is zero.

                 AUTORUN_SCHEDULE * * * *       :- Automatic run at specific time in daemon mode.
                                  - - - -
                                  Â¦ Â¦ Â¦ Â¦
                                  Â¦ Â¦ Â¦ +----- day of week (0 - 6) (0 to 6 are Sunday to Saturday)
                                  Â¦ Â¦ +---------- month (1 - 12)
                                  Â¦ +--------------- day of month (1 - 31)
                                  +-------------------- hour (0 - 23)

                     example: orachk -set 'AUTORUN_SCHEDULE=8,20 * * 2,5' will schedule runs on tuesday and friday at 8 and 20 hour.

                 AUTORUN_FLAGS <flags> : orachk flags to use for auto runs.

                     example: orachk -set 'AUTORUN_INTERVAL=12h;AUTORUN_FLAGS=-profile sysadmin' to run sysadmin profile every 12 hours

                              orachk -set 'AUTORUN_INTERVAL=2d;AUTORUN_FLAGS=-profile dba' to run dba profile once every 2 days.

                 NOTIFICATION_EMAIL : Comma separated list of email addresses used for notifications by daemon if mail server is
configured.

                 PASSWORD_CHECK_INTERVAL <number of hours> : Interval to verify passwords in daemon mode

                 COLLECTION_RETENTION <number of days> : Purge orachk collection directories and zip files older than specified days.
        [-id <ID>] -unset <parameter | all>
                unset the parameter
                if '-id <ID>' is specified, it will unset the parameter for specified autorun schedule ID
                  example: orachk -unset AUTORUN_SCHEDULE

        [-id <ID>] -get <parameter | all>
                Print the value of parameter.
                if '-id <ID>' is specified,  it will print the value of parameter for specified autorun schedule ID



        -vmguest
                Pass comma separated filenames containing exalogic guest VM list(applicable for Exalogic only)

        -hybrid [-phy]
                phy     :Pass comma separated physical compute nodes(applicable for Exalogic only)
                         eg:- ./orachk -hybrid -phy phy_node1,phy_node2


        Profile Run Options:
        -profile
                Pass specific profile.
                With -h prints help.
                List of supported profiles:

                 asm            asm Checks
                 clusterware    Oracle clusterware checks
                 compute_node   Compute Node checks (Exalogic only)
                 control_VM     Checks only for Control VM(ec1-vm, ovmm, db, pc1, pc2). No cross node checks
                 dba            dba Checks
                 ebs             Oracle E-Business Suite checks
                 el_extensive    Extensive EL checks
                 el_lite         Exalogic-Lite Checks(Exalogic Only)
                 el_rackcompare  Data Collection for Exalogic Rack Comparison Tool(Exalogic Only)
                 emagent         Cloud control agent checks
                 em              EM related checks
                 goldengate      Oracle GoldenGate checks
                 maa             Maximum Availability Architecture Checks
                 nimbula         Nimbula checks for Exalogic
                 obiee           obiee Checks(Exalytics Only)
                 platinum        Platinum certification checks
                 preinstall      Pre-installation checks
                 prepatch        Checks to execute before patching
                 storage         Oracle Storage Server Checks
                 switch          Infiniband switch checks
                 sysadmin        Sysadmin checks
                 timesten        Timesten Checks(Exalytics Only)
                 virtual_infra   OVS, Control VM,  NTP-related and stale VNICs check (Exalogic Only)
                 zfs             ZFS storage appliances checks (Exalogic Only)

        -excludeprofile
                Pass specific profile.
                List of supported profiles is same as for -profile.


        -cells
                Pass comma separated storage server names to run orachk only on selected storage servers.

        -ibswitches
                Pass comma separated infiniband switch names to run orachk only on selected infiniband switches.

        -zfsnodes
                Pass comma separated ZFS storage appliance names to run orachk only on selected storage appliances.

        -dbserial
                Run SQL, SQL_COLLECT and OS Checks in serial

        -dbparallel [n]
                Run SQL, SQL_COLLECT and OS Checks in parallel.
                n       : Specified number of Child processes. Default is 25% of CPUs.
```

Guidelines for Running ORAchk

1. If the oracle user exists on the system and all the Oracle components are installed or running (CRS, RDBMS, ASM) it is recommended to run ORAchk as the oracle (or RDBMS software install) user.   The tool will need to perform some collections that require root privilege in which case it will display the following output (or similar):

```
. . .
40 of the included audit checks require root privileged data collection. If sudo is not
configured or the root password is not available, audit checks which require root privileged data
collection can be skipped.

1. Enter 2 if you will enter root password for each host when prompted
2. Enter 1 if you have sudo configured for oracle user to execute root_orachk.sh script
3. Enter 3 to skip the root privileged collections
4. Enter 4 to exit and work with the SA to configure sudo or to arrange for root access and run
the tool later.
. . .
```

Note: It is recommended to run the tool as the database software owner (e.g. oracle). The user may run the tool as the Grid Infrastructure software owner (e.g. grid) and it will collect the same data but database credentials must manually be supplied to perform the database related audit checks. Typically when run as oracle the customer will have OS authentication set up for the oracle database software owner and the database login credentials will not be needed. If the clusterware and database are installed and running it is best to run the tool as oracle (or whatever user owns the oracle database software installation). If for some reason you want to run the tool as another user then refer to VERIFYING DATABASE AUTHENTICATION in the TROUBLESHOOTING section for additional information on how to verify database authentication before running the tool.

2. The user should only use the driver files which are distributed with the tool kit and the driver files should not be edited by hand.   A new version of the tool requires new driver files.

How to Run ORAchk Interactively

1. Log in to the system as the *Oracle RDBMS software installation owner* (if Oracle products installed, otherwise log in as root) -- See Usage Considerations for details

2. Stage the appropriate orachk.zip kit in its own directory the node on which the tool will be executed

3. Unzip orachk.zip kit, leaving the script and driver files together in the same directory

4. Validate the permissions for orachk are 755 (-rwxr-xr-x). If the permissions are not currently set to 755, set the permissions on orachk as follows:

```
$ chmod 755 orachk
```

5. Invoke the tool as follows:

```
$ ./orachk
```

   Follow the prompts while reading and understanding all messages. The Q&A process of ORAchk will be similar to that shown below:

```
CRS stack is running and CRS_HOME is not set. Do you want to set CRS_HOME to
/oragi/11.2.0.2?[y/n][y]y

Checking ssh user equivalency settings on all nodes in cluster

Node ratgen02 is configured for ssh user equivalency for oracle user


Searching for running databases . . . . .

. . .


ORAchk found database cosp registered in OCR. Is this the database to check best practices
for?[y/n][y]y

Checking Status of Oracle Software Stack - Clusterware, ASM, RDBMS

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
----------------------------------------------------------------------------------------------------------
Oracle Stack Status
----------------------------------------------------------------------------------------------------------
Host Name CRS Installed ASM HOME RDBMS Installed CRS UP ASM UP RDBMS UP DB Instance Name
----------------------------------------------------------------------------------------------------------
ratgen01 Yes Yes Yes Yes Yes Yes maadb1
ratgen02 Yes Yes Yes Yes Yes Yes maadb2
----------------------------------------------------------------------------------------------------------


This script relies upon the environment being properly set
```

```
Please examine the above environment settings. Are they correct?[y/n][y]y

9 of the included audit checks require root privileged data collection . If sudo is not
configured or the root password is not available, audit checks which  require root privileged
data collection can be skipped.


1. Enter 1 if you will enter root password for each  host when prompted

2. Enter 2 if you have sudo configured for oracle user to execute root_orachk.sh script

3. Enter 3 to skip the root privileged collections

4. Enter 4 to exit and work with the SA to configure sudo  or to arrange for root access and run
the tool later.
```

> **Note:** If you chose option 1, to provide root password when prompted, you will be prompted once for each node during the data collection phase for the nodes (unless the EXPECT utility is installed, in which case you will only be prompted once). If you do not enter the root password in a timely way (within RACCHECK_TIMEOUT) then the root privileged collections and audit checks for that node will be skipped.
>
> Starting with ORAchk 2.2.2, the sysadmin profile may be run independently as the root user allowing for the root checks to be performed independently of the non-root checks (./orachk -profile sysadmin).

```
Please indicate your selection from one of the above options[1-4][3]:- 2

*** Checking RAC Best Practice Recommendations (PASS/WARNING/FAIL) ***



Performing SQL collections for use with audit checks on cosp...please stand by.

. . . . . . . . . . . . . . .
```

6. Upon completion, the following (or similar) will be displayed:

```
Detailed report (html) -
/export/home/oracle/exachk/exachk_scai02db01_012714_143909/exachk_hostname1_012714_143909.html


UPLOAD(if required) - /export/home/oracle/orachk/orachk_hostname1_012714_143909.zip
```

7. At this point you may view the HTML report shown in the output above. If there is an active SR which ORAchk was recommended as part of its resolution, upload the orachk_*.zip to that SR.

How to Run ORAchk Silently

Note: Following configuration is required only if customer does not want to use orachk daemon functionality. Check "ORAchk Daemon Mode Operation" section for more detail about orachk daemon.

ORAchk can be optionally run in "silent" or "non-interactive" mode in order to enable scheduling and automation. There are specific pre-requisites that must be met in order to execute ORAchk in silent mode. For this reason, this section is broken out into sub-sections, ORAchk Silent Pre-requisites and Running ORAchk Silently.

Pre-requisites

1. In order to execute ORAchk silently SSH user equivalence is mandatory. That said, you must configure SSH user equivalence for RDBMS software owner (e.g. oracle) between the system that will be used for the actual execution of ORAchk and all other cluster nodes. For instructions on configuring user equivalence see My Oracle Support Note: 372795.1. Once configured, validate the proper configuration of SSH user equivalence configuration as the RDBMS software owner (e.g. oracle) by executing following command replacing "<dbServerName>" with the appropriate remote cluster node:

```
$ ssh -o NumberOfPasswordPrompts=0 -o StrictHostKeyChecking=no -l oracle <dbServerName> "echo
\"oracle user equivalence is setup correctly\""
```

   If the result of the above command is similar to "Permission denied (publickey,gssapi-with-mic,password)" then the SSH user equivalence is not properly configured. Consult My Oracle Support Note: 372795.1 for details on configuring SSH user equivalence.

   Note: SSH user equivalence is ONLY necessary for those systems running Grid Infrastructure for supporting a cluster, this is NOT required for Single Instance Databases or Oracle Restart Configurations.

2. As previously mentioned, in order to take advantage of the full functionality of ORAchk, root access is required to run the root specific checks. To facilitate these checks in silent mode we MUST configure passwordless SUDO and utilize the -s (more on this later) option to execute ORAchk. Instructions for configuring SUDO for use by ORAchk are found in Appendix M.

   Note: If sudo is not allowed within your environment, this step may be skipped. When in this situation you can still execute ORAchk silently without the root specific checks using the -S option (more on this later). Eliminating the root specific checks limits the capabilities of ORAchk and is therefore not the recommended method of executing ORAchk.

Running ORAchk Silently/Non-Interactively

1. Log in to the system as the *Oracle RDBMS software installation owner* (if Oracle products installed, otherwise log in as root) -- See Usage Considerations for details

2. Stage the appropriate orachk.zip kit in its own directory the node on which the tool will be executed

3. Unzip orachk.zip kit, leaving the script and driver files together in the same directory

4. Validate the permissions for orachk are 755 (-rwxr-xr-x). If the permissions are not currently set to 755, set the permissions on orachk as follows:

```
$ chmod 755 orachk
```

5. To run ORAchk silently you will need to specify one of the following arguments depending on your configuration:
   -s - unattended execution when the Oracle user can SUDO to root without a password
   -S - unattended execution without the root password and root privileged collections/audits

   Note: It is highly recommended to implement passwordless SUDO to root for /$HOME/root_orachk.sh for full functionality of the ORAchk utility. See Appendix M for details on how to configure SUDO for ORAchk.

   Assuming SUDO has been properly configured ORAchk (See Appendix M) can now be invoked silently as follows:

   Note: If SUDO has not been configured, the -S option must be specified in place of the -s option.

```
$ ./orachk –s
```

   ORAchk will now be executed in silent mode with the information it has pulled from the Clusterware. When in silent mode, all databases running on the local node that are defined as Clusterware resources will have data collection and audit checks performed.

6. Upon completion, the following (or similar) ORAchk output will be available for review:

```
Detailed report (html) -
/export/home/oracle/exachk/exachk_scai02db01_012714_143909/exachk_hostname1_012714_143909.html


UPLOAD(if required) - /export/home/oracle/orachk/orachk_hostname1_012714_143909.zip
```

   Note: If there is an active SR which ORAchk was recommended as part of its resolution, upload the orachk_*.zip produced by the latest run of the tool to that SR.

ORAchk Daemon Mode Operation

Starting ORAchk Daemon Mode Interactively

ORAchk version 2.2.2 introduces the daemon process functionality to permit non-interactive (batch or silent mode) execution on a regular interval.

Note: When running ORAchk in daemon mode, the most recent and next most recent (if any) collection reports are automatically compared. If the NOTIFICATION_EMAIL address is configured a summary will be emailed along with attachments for the reports and the comparison report.

Before running ORAchk in daemon mode it is recommended to customize the daemon parameters (using the –set flag):

- AUTORUN_INTERVAL – defines the interval in which ORAchk will be executed, specified in days or hours (<d|h>).
- PASSWORD_CHECK_INTERVAL – defines the frequency (specified in hours) in which the running daemon validates the passwords entered when the daemon was started. Should an invalid password be found (due to a password change), the daemon will halt and notification will be provided via the daemon log (orachk_daemon.log) and email (if configured).
- NOTIFICATION_EMAIL – allows for emailing of notifications produced by the ORAchk daemon. You can have multiple email address separated by comma.
- AUTORUN_FLAGS – allows for the setting of ORAchk execution parameters for auto-run. The available parameters are listed in the *Usage* section of this document.
- COLLECTION_RETENTION:- files created by the orachk daemon by executing a scheduled orachk run can be cleaned if they are older than the collection_retention number of days.
- INFO:- orachk –d info command will display information about the orachk daemon eg., where it is being run from and when was it started etc.
- STOP_CLIENT:- when you stop the orachk daemon and if an orachk client run as in progress, the orachk daemon will continue to run. Using the stop_client option with –d flag will stop the current client run in progress and then will stop the orachk daemon so the user does not have to wait until the client run is finished

More than one parameter can be set in a single execution by providing each parameter in a semi-colon delimited list. To set the daemon to run every day, in verbose mode, specifying an email address for daemon notices and check for changed passwords hourly, execute the following command as the userid that will launch ORAchk in daemon mode (all on one line):

```
$ ./orachk -set "AUTORUN_INTERVAL=1d;AUTORUN_FLAGS= -o
v;NOTIFICATION_EMAIL=firstname.lastname@company.com;PASSWORD_CHECK_INTERVAL=1"
```

To see the current ORAchk daemon values, as the userid that launched the daemon execute "./ORAchk -get <parameter_name> | all":

```
$ ./orachk –get  AUTORUN_INTERVAL

AUTORUN_INTERVAL = 1d

$ ./orachk -get all

AUTORUN_INTERVAL = 1d
AUTORUN_FLAGS = -o v
NOTIFICATION_EMAIL = firstname.lastname@company.com
PASSWORD_CHECK_INTERVAL = 1
```

ORAchk version 2.2.3 introduces the ability to run ORAchk on a specific schedule, using the "AUTORUN_SCHEDULE" feature. There are four fields expected after this keyword. The first is for the hour of the day, with 0-23 the valid values. The second is for the day of month, with 1-31 the valid values. The third is for the month, with 1-12 the valid values. The fourth is a specific day of the week, with 0-6 the valid values. The fields can be either single values, or comma separated. For example to set the ORAchk daemon to run on hours 15,16,and 17 of every day, of every month, on Tuesday.:

```
$ ./orachk -set "AUTORUN_SCHEDULE=15,16,17 * * 2"
```

The parameters can be also be set or modified after the ORAchk daemon process has started.

> **Note:** It is highly recommended that at minimum NOTIFICATION_EMAIL and PASSWORD_CHECK_INTERVAL be configured.

After the appropriate ORAchk daemon parameters have been set to configure the daemon, start the ORAchk daemon process:

```
$ ./orachk –d start
```

ORAchk will present the standard interactive graphical user interface to collect the required information and start the daemon process. When the input collection process has completed, you should see output similar to:

```
Daemon is started with PID : 27952
```

The current status of the ORAchk Daemon may be checked by executing the following:

```
$ ./orachk –d status
Daemon is running. PID : 27952
```

The ORAchk Daemon may be stopped at any time by executing the following:

```
$ ./orachk –d stop
```

Once the ORAchk daemon is running, ORAchk will automatically run in accordance to the AUTORUN_INTERVAL specified. The next collection time can be queried from the daemon as follows:

```
$ ./orachk -d nextautorun

Next auto run starts on Jun 10, 2013 11:26:55
```

ORAchk may also be executed on demand (user initiated) through the daemon by simply executing “orachk” without any arguments as the user that started the daemon process from the same directory from which the ORAchk daemon process was launched. This execution is non-interactive (daemon passes parameters to all prompts) but will produce output on the screen similar to standard execution:

> **Note:** If you do have the ORAchk Daemon running and wish to execute ORAchk in standard interactive mode, you may specify the “-nodaemon” flag.

```
$ ./orachk
Sending commands to daemon (mypid 27594) args :

CRS stack is running and CRS_HOME is not set. Do you want to set CRS_HOME to
```

```
/u01/app/11.2.0/grid?[y/n][y]y

...
```

The ORAchk daemon that was started by a given user CANNOT be used to run ORAchk by a different user, e.g. if user1 started daemon then user2 cannot use ORAchk daemon to run ORAchk in non-interactive mode. In fact to use ORAchk daemon, you will have to be same user and will have to start on-demand run from same directory where you started daemon.

Once started, the ORAchk Daemon will continue to run in the background until an explicit stop of the daemon (orachk –d stop) or one of the following conditions is met:

- The Server in which the daemon is running is rebooted or goes down.
- If password is changed on any node, daemon will stop and an entry will be placed in ORAchk_daemon.log as well as send an email to NOTIFICATION_EMAIL. The PASSWORD_CHECK_INTERVAL parameter can be tuned to ensure validity of the required passwords.
- If ORAchk script has changed or replaced with new script since you started daemon, further on-demand as well as auto run will not succeed. You will have to restart daemon with new script for future run.

Note: If the system configuration changed (e.g. add/delete nodes, add/delete instances, etc), you will have to restart ORAchk daemon for the configuration changes to be recognized

Auto-starting ORAchk Daemon Mode Non-Interactively

In orachk 2.2.4, an optional daemon mode auto-start functionality is provided. This feature requires configuration of passwordless ssh user equivalence to root for the user configuring the auto-start feature (eg., root or oracle). If passwordless ssh user equivalence for the user running orachk is not found, orachk will be able to configure it at the option of the user. This passwordless ssh user equivalence is left in place as long as the orachk auto-start functionality is configured. De-configuring the orachk daemon mode auto-start feature will restore the ssh configuration to the state it was found before the auto-start feature was configured. This feature is useful if there is a requirement for orachk to start automatically in daemon mode non-interactively after a node is rebooted for any reason.

There are three new options for the orachk daemon auto-start configuration:

1. -initcheck:- To check whether the orachk daemon is configured to auto-start at node reboot or if the daemon goes down for some reason

2. -initsetup:- To configure the orachk daemon auto-start mode via init integration with init.orachk script

3. -initrmsetup:- To remove the orachk daemon auto-start mode and int integration with init.orachk script

Configuring Multiple Runs within Daemon

ORAchk 2.2.5 introduced the ability to have multiple runs defined in the ORAchk daemon. For example, a configuration to run the “sysadmin” profile on a certain schedule, and a configuration to run the “dba” profile on a different schedule, both within the same daemon.

To keep configurations separate, 2.2.5 introduced the “-id <ID>” command line qualifier. This is required when using multiple configurations, and it identifies the separate configurations.

In this example, we will create two configurations within the same daemon, one to run the sysadmin profile, and another to run the dba profile.

ORAchk has been installed and we have cd’d into the installation directory. This is not a role separated environment, the “oracle” userid owns both the grid and RDBMS homes.

Validate that the daemon is not configured or running:

```
$ ./orachk –get all
None of parameters are set

$ ./orachk –d status
Orachk daemon is not running.
```

1. Configure the first “ID”:

```
$ ./orachk -id DBA_PROF -set \
> "NOTIFICATION_EMAIL=my.email@oracle.com;\
> AUTORUN_SCHEDULE = 4,8,12,16,20 * * *;\
> AUTORUN_FLAGS=-profile dba;\
> COLLECTION_RETENTION=30"

Created NOTIFICATION_EMAIL for ID[DBA_PROF]

Created AUTORUN_SCHEDULE for ID[DBA_PROF]

Created AUTORUN_FLAGS for ID[DBA_PROF]

Created COLLECTION_RETENTION for ID[DBA_PROF]
```

2. Configure the second “ID”

```
$ ./orachk -id SYSADMIN_PROF -set \
> "NOTIFICATION_EMAIL=my.email@oracle.com;\
> AUTORUN_SCHEDULE = 6,10,14,18,22 * * *;\
> AUTORUN_FLAGS=-profile sysadmin;\
> COLLECTION_RETENTION=30"

Created NOTIFICATION_EMAIL for ID[SYSADMIN_PROF]

Created AUTORUN_SCHEDULE for ID[SYSADMIN_PROF]

Created AUTORUN_FLAGS for ID[SYSADMIN_PROF]

Created COLLECTION_RETENTION for ID[SYSADMIN_PROF]
```

3. Review the configuration:
   Note: Each profile can be viewed individually by specifying the –id <ID> option.

```
$ ./orachk -get all
```

```
ID: DBA_PROF
----------------------------------
NOTIFICATION_EMAIL = my.email@oracle.com
AUTORUN_SCHEDULE = 4,8,12,16,20 * * *
AUTORUN_FLAGS = -profile dba
COLLECTION_RETENTION = 30

ID: SYSADMIN_PROF
----------------------------------
NOTIFICATION_EMAIL = my.email@oracle.com
AUTORUN_SCHEDULE = 6,10,14,18,22 * * *
AUTORUN_FLAGS = -profile sysadmin
COLLECTION_RETENTION = 30
```

4. Finally Start the Daemon:

```
$ ./orachk -d start

CRS stack is running and CRS_HOME is not set. Do you want to set CRS_HOME to
/u01/app/11.2.0.3/grid?[y/n][y]

Checking ssh user equivalency settings on all nodes in cluster

Node cetrain02 is configured for ssh user equivalency for oracle user

Searching for running databases . . . . .

. . . .
List of running databases registered in OCR
1. mydb
2. None of above

Select databases from list for checking best practices. For multiple databases, select 1 for All
or comma separated number like 1,2 etc [1-2][1].
. . .

<output truncated for brevity>

Verifying nm2user password.

. . .

orachk daemon is started with PID : 25872
```

Running subset of checks using profile

There are three reasons to use profiles in ORAchk

1. Running subset of checks:- if user just want to execute only clusterware related checks, rather than running full healthcheck and ignoring rest of the checks, one can use clusteware profile like ./orachk –profile clusterware and it will execute clusterware related checks only.
2. Running orachk as root:- Even oracle databases are up and running, one can run orachk as root user also as long as they use syadmin profile like ./orachk –profile sysadmin
3. Support role based health checks:- you might have different people for OS and database management and might execute separate health checks. Using sysadmin profile one can do OS health check and dba can do database health check using dba profile

Performing Report Comparisons with ORAchk

As of version 2.2.1, ORAchk has the ability to perform report comparisons between 2 ORAchk reports. This allows for trending of Success Factor and Best Practice changes over time, after planned maintenance, etc within a user friendly HTML report. The pre-requisite for doing so is to maintain the

ORAchk output directories, .zip output files or the ORAchk HTML reports of those ORAchk runs that will be used in the report comparisons. The steps below provide an example performing a ORAchk report comparison:

1. Determine the reports (old and new) to be used in the report comparison:
   Note: The example uses the output directories however; the HTML reports or output .zip files may also be used for the comparison feature.

```
$ ls -ald orachk_cetrain01*
drwxr-xr-x. 2 oracle oinstall  45056 Feb  8 12:38 orachk_cetrain01_ORCL_020813_105850
drwxr-xr-x. 2 oracle oinstall  45056 Feb 12 13:27 orachk_cetrain01_ORCL_021213_131815
drwxr-xr-x. 2 oracle oinstall  36864 Feb 27 16:41 orachk_cetrain01_ORCL_022713_163238
```

2. For our report comparison we will use the Feb 8th report and the Feb 27th report:

```
$ ./orachk -diff orachk_cetrain01_ORCL_020813_105850
orachk_cetrain01_ORCL_022713_163238
Summary
Total   : 157
Missing : 12
New     : 0
Changed : 7
Same    : 138
File comparison is complete. The comparison report can be viewed in:
/home/oracle/orachkbeta/orachk_020813105850_022713163238_diff.html
```

Note: When running ORAchk in daemon mode, the most recent and next most recent (if any) collection reports are automatically compared. If the NOTIFICATION_EMAIL address is configured a summary will be emailed along with attachments for the reports and the comparison report.

Merging ORAchk Reports

There may be times when you have multiple reports of orachk and you want to merge them into a single report to review. For example,

1. Due to customer security policies passwordless ssh user equivalence between cluster nodes is not allowed making it necessary to run orachk locally each cluster node. The reports from each node can be merged into a single report.
2. In cases where the DBA does not know the root password and runs orachk using the dba profile and the sysadmin does know the root password and runs orachk as root using the sysadmin profile. The two reports can be merged into a single report.

   To merge reports. simply pass the list of orachk zip files or orachk output directories containing the reports to be merged and orachk will the merge the independent reports into a single report.

   Eg., $ ./orachk –merge <zipfile 1> <zip file 2> > <zip file 3> > <zip file …>

Running ORAchk in Upgrade Readiness Mode

Starting with Version 2.1.5, ORAchk (Oracle Configuration Audit Tool) can be used to obtain an automated 11.2.0.3 (or above) Upgrade Readiness Assessment. The goal of the ORAchk Upgrade Readiness Assessment is to make the process of upgrade planning for Oracle RAC and Oracle Clusterware target versions 11.2.0.3 and above as smooth as possible by automating many of the manual pre and post checks detailed in upgrade related documents.

ORAchk Upgrade Readiness has 2 modes, a pre-upgrade check and a post-upgrade check. The pre-upgrade check is to be executed during the planning phase of the upgrade process; this will ensure that enough time is available to rectify potential issues prior to the upgrade itself. The post-upgrade mode ensures the health of GI and Database (from an Upgrade perspective). Below is a summary list of what to expect from ORAchk Upgrade Readiness:

- The target Clusterware and database versions must be 11.2.0.3 or above
- In pre-upgrade mode the tool will detect all databases registered in the clusterware automatically and present a list of databases on which to perform pre-upgrade checks.
- In post-upgrade mode the tool will detect all databases registered in the clusterware automatically and present a list of databases on which to perform post-upgrade checks. If any pre-11.2.0.3 databases are selected, the post-upgrade checks will be skipped for them.
- In both modes the tool will check the clusterware and OS appropriately.
- When the tool completes the user will be referred to an HTML formatted report which will contain the findings and links to additional details and information.

The steps below provide an example performing an ORAchk Upgrade Readiness Assessment:

1. During the upgrade planning phase of your pending RAC upgrade, execute ORAchk in pre-upgrade mode as the Oracle RDBMS software owner and follow the on-screen prompts:
   Note: It is HIGHLY recommended that the pre-upgrade checks are executed in the planning phase of the upgrade. This will allow planning and implementation of findings highlighted by ORAchk.

```
$ ./orachk -u -o pre
CRS stack is running and CRS_HOME is not set. Do you want to set CRS_HOME to /u01/crs?[y/n][y]y

. . .

Detailed report (html) - /home/oracle/orachk/orachk_
ratlnx01_ORCL_022813_131302/orachk_cetrain11_ORCL_022813_131302.html


UPLOAD(if required) - /home/oracle/orachk/orachk_ratlnx01_ORCL_022813_131302.zip
```

2. Once you have successfully upgraded to 11.2.0.3 or higher (GI and/or RDBMS), execute ORAchk in post-upgrade mode as the Oracle RDBMS software owner and follow the on-screen prompts:

```
$ ./orachk -u -o post
CRS stack is running and CRS_HOME is not set. Do you want to set CRS_HOME to
/u01/app/11.2.0/grid??[y/n][y]y
```

```
. . .

Detailed report (html) -
/home/oracle/orachkbeta/orachk_cetrain11_ORCL_022813_132849/orachk_cetrain11_ORCL_022813_132849.h
tml


UPLOAD(if required) - /home/oracle/orachk/orachk_cetrain11_ORCL_022813_132849.zip
```

Other Useful Options

- dbnone:- Introduced in version 2.2.4 this option does accept any value. If this option is specified at the command prompt while running orachk the user will not be prompted to select any running databases and no database checks will be run.
- dball:- Introduced in version 2.2.4, this option also does not accept any value. If this option is specified at the command prompt while running orachk, the user will not be prompted but ALL running databases will be checked
- dbnames:- Introduced in version 2.2.4, this option requires a list of databases to be passed at the command prompt. The user will not be prompted to select any other running databases. This option is similar in effect to the RAT_DBNAMES environemt variable discussed below in the Local Environmental Issues section but is specified at the command line rather than in the environment.
- nopass:- Introduced in version 2.2.4, this option also does not take any argument. If this option is used, checks which PASS will not be included in the HTML report, nor will they be uploaded to the database if the database upload functionality is configured according to Appendix F - Uploading ORAchk Results and Patches to a Database for Reporting**.**
- noscore:- Introduced in version 2.2.4, the noscore option allows the user to omit the health score which appears at the top of the ORAchk HTML report

Appendix A - Troubleshooting

Debugging ORAchk

To produce debug output for ORAchk, simply execute ORAchk with the "-debug" command line option. ORAchk will create a debug file whose name is based on a timestamp, including:.

a. bash -x of program on local node
b. bash -x of program on all remote nodes
c. bash -x of all dynamically generated and called scripts in orachk

Debugging ORAchk Daemon

Note: Each client run will generate a log in the working directory. The amount of information inside this client run log depends upon whether the daemon is running in debug or non debug mode.

1. Start_debug:- To troubleshoot daemon issues, the user can start the orachk daemon in debug mode using the start_debug option with –d flag. Daemon debugging information will be written in the orachk_daemon_debug.log generated in the orachk working directory. If the daemon is started in debug mode then client log will contain:
   a. 'bash –x' of program on local node
   b. 'bash –x' of program on all remote nodes
   c. bash -x of all dynamically generated and called scripts in orachk

2. If the daemon is started in non debug mode (orachk -d start) then the client log will contain normal program execution . This will be beneficial in debugging daemon related issues especially password checking issues and client run submission and client run related issues (local node, remote node and intermediary scripts).

Environment Variables for Controlling Behavior

1. Runtime Command Timeouts:

   RAT_TIMEOUT - default 90 seconds, if the default timeout for any non-root privileged individual commands is not long enough then the watchdog daemon will kill that child process and the desired data will be missing. If that is happening the timeout can be lengthened by setting this environment variable in the script execution environment:

```
$ export RAT_TIMEOUT=120
```

   RAT_ROOT_TIMEOUT - default 300 seconds, the tool will execute a set of root privileged data collections once for each node in the cluster. If the default timeout for the set of root privileged data collections is not long enough then the watchdog daemon will kill that child process and the desired data will be missing for that node. If that is happening the timeout can be lengthened by setting this environment variable in the script execution environment:

```
$ export RAT_ROOT_TIMEOUT=600
```

Note: If any of these timeouts is being encountered it is best to determine the cause of the delay and correct it. Run the tool during times of least load on the system. Missing data collections will limit the value of the tool.

2. The orachk_error.log has errors:

   When examining the orachk_error.log for the tool some errors will most likely be listed. Some of those are expected errors and are not indicative of any problem. These errors are redirected and "absorbed" into the error.log to keep them from being reported to the screen and cluttering the display. So do not report any of the following errors to Support.

   For instance, an error similar to the following may be reported numerous times (once for each Oracle software home for each node):

```
/bin/sh: /u01/app/11.2.0/grid/OPatch/opatch: Permission denied
chmod: changing permissions of `/u01/app/oracle_ebs/product/11.2.0.2/VIS_RAC/.patch_storage':
Operation not permitted
OPatch could not open log file, logging will not be possible
Inventory load failed... OPatch cannot load inventory for the given Oracle Home.
```

   These types of errors occur in role separated environments when the tool which is run as the RDBMS software owner attempts to list the patch inventories of homes owned by other users (grid, or other database home owners) using opatch. So when opatch is invoked to list the patch inventories for those other users it errors out because the current user do not have permissions on the other homes. In these cases the opatch error is ignored and the patch inventories for those homes are gathered by other means.

   Additionally, errors similar to the following are ignorable:

```
./orachk: line [N]: [: : integer expression expected
```

   The line number may change over time but this error just means we were expecting an integer return value and no value was found (i.e., empty) and so the shell returns that error in trying to make the comparison. This error could be repeated many times for the same command, once for each node.

3. Remote root Login Problems

   If remote root login is not permitted over SSH the root privileged commands will not work in case of option 2 explained above. To verify oracle software owner (e.g. oracle) the user should execute the following command manually from whichever node it's not working and make sure he gets output as follows. If remote root login is not working then the tool will not be able to check the remote nodes. Please engage the systems administrators to correct this if only temporarily for running the tool.

   How to verify expected output if remote root login over SSH is configured:

```
$ ssh root@remotehostname "id"
root@remotehostname's password:
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

If the remote root login can be configured edit the /etc/ssh/sshd_config file as follows:

```
PermitRootLogin to yes
```

Now execute the following command as root on all nodes of the cluster.

```
# /etc/init.d/sshd restart
```

4. Local Environmental Issues

   The utility attempts to derive all the data it needs from the environment (OS and OCR) however it might be possible at times that the tool doesn't work as expected due to local system variances and it is difficult to anticipate and test every possible scenario. Therefore support for a number of environment variables have been included to provide a way for the user to over-ride the default behavior of the utility or to give it the information it needs in order to work as expected. The list of variables and their usage is as follows:

   RAT_INV_LOC - if the oraInst.loc file is not where expected, e.g. /u01/app/oraInventory, the user may need to specify the exact location of the oraInventory directory.

   RAT_CRS_HOME - Set this variable if the utility cannot derive the correct CRS_HOME path. The user can tell if this variable is required if he/she knows that the clusterware is installed but the utility thinks it is not.

   RAT_ORACLE_HOME - Set this variable if the utility cannot derive the correct ORACLE_HOME paths for the databases registered in the clusterware. The user will be able to determine if this variable is required if they know the database software is installed but the utility displays information to the effect that the software is not installed. The tool will perform best practice and recommended patch checks for all the databases running from the home specified in RAT_ORACLE_HOME.

   RAT_ASM_HOME - if the utility cannot derive the correct ASM_HOME path from the clusterware. The user will be able to determine if this variable is required if they know the ASM software is installed in a separate home but the utility displays information to the effect that the software is not installed.

   RAT_OS - if the utility improperly derives the platform needed. The utility will prompt the user that the data needed for the derived platform could NOT be found due to improperly detecting an unsupported platform.

   RAT_DB - if the utility improperly derives an incorrect database version

   RAT_DBNAMES - if the utility doesn't derive valid database names from the clusterware a space delimited list of databases can be specified and the tool will use that list instead of what it derives from the clusterware, e.g., export RAT_DBNAMES="ORCL ORADB PROD". Only use double quotes if specifying more than one database. Note that if you configure RAT_DBNAMES as a subset of databases registered in the clusterware, and you want the patch inventories of ALL databases found registered in the clusterware to have their patch inventories checked for recommended patches then you should also configure RAT_DBHOMES.

RAT_DBHOMES - If RAT_DBNAMES is set then by default the recommended patch analysis will be limited to the homes for the list of databases specified in RAT_DBNAMES. If it is desirable to perform the recommended patch analysis for additional database homes than those specified in RAT_DBNAMES then specify the space delimited list of databases whose homes should be checked for recommended patches, e.g., assume export RAT_DBNAMES="ORCL ORADB" but you also want the PROD database home, even if the PROD database is down, to be checked then export RAT_DBHOMES="ORCL ORADB PROD". In this way, best practices will be checked for the ORCL and ORADB databases but the recommended patches will be checked for the homes of ORCL, ORADB and PROD. Only use double quotes if specifying more than one database.

RAT_SSHELL – overrides the default secure shell location in case ssh is not where expected (i.e., /usr/bin/ssh). If ssh commands return the error "-bash: /usr/bin/ssh -q: No such file or directory " then it may be because ssh isn't located where expected and this variable can be used to redirect the tool to the proper location.

RAT_SCOPY - overrides the default secure copy location in case scp is not where expected (i.e., /usr/bin/scp). If scp commands return the error ”/usr/bin/scp -q: No such file or directory “ then it may be because scp isn't located where expected and this variable can be used to redirect the tool to the proper location.

5. Linux “expect” Utility Troubleshooting

ORAchk uses the “expect” utility when available to answer password prompts to connect to remote nodes for password validation as well as executing root collections, without logging the actual connection process by default. There are two ORAchk environment variables that may be set to help debug remote target connection issues.

RAT_EXPECT_DEBUG - If this variable is set to “-d” (dash d, not underscore d), expect command tracing will be activated, and the trace information is written to standard output. For example, “export RAT_EXPECT_DEBUG=-d”.

RAT_EXPECT_STRACE_DEBUG – If this variable is set to “strace”, strace calls the expect command, and the trace information is written to standard output. For example, “export RAT_EXPECT_STRACE_DEBUG=strace”.

By varying the combinations of these two variables, three levels of expect connection tracing are available.

Note: These two variables should only be set at the direction of Oracle support or development. They are typically used in combination with other variables and user interface options to restrict the amount of data collected during the tracing, and the “script” command to capture standard output. They should not be set for a full orachk run as that will generate a large amount of data, and if the “script” command is not used, the trace data will simply scroll by on the screen and be lost!

6. Database Login Problems:

   VERIFYING DATABASE AUTHENTICATION - if you intend to run the tool as a user other than the database software installation owner (root or grid) and if you experience problems connecting to the database then perform the following steps:

```
1.  login as grid(OS) user on the system.
2.  export ORACLE_HOME=<path of Oracle database home>
3.  export ORACLE_SID=<database SID>
4.  export PATH=$ORACLE_HOME/bin:$ORACLE_HOME/lib:$PATH
5.  add alias in $ORACLE_HOME/network/admin/tnsnames.ora for <database SID>.
6.  connect to database using $ORACLE_HOME/bin/sqlplus "sys@<SID> as sysdba" and enter
    password.
7.  Make sure you have successful connection.
```

   If this method of connecting to the database will not work then the tool will not be able to connect either. Consider running the tool as the Oracle database software installation owner.

7. User Profiles:

   The presence of prompts (read -p) or traps in the user profile can lead to the tool hanging during execution as the tool sources the .profile at runtime. For that reason the tool checks the .profile in the environment for these statements and presents a message telling the user to temporarily comment those statements on all nodes.

ORAchk on SuSe Reports “-bash: ./orachk: /bin/env: bad interpreter: No such file or directory”

   On SuSe, prior to executing ORAchk you must softlink /usr/bin/env to /bin/env (ln -s /usr/bin/env /bin/env) on all nodes.

Appendix B - How to Obtain Support

If problems are encountered either at runtime or if there are questions about the content of the findings of the tool, please post your issues/questions/concerns to the ORAchk thread within the ORAchk Thread of the Scalability RAC My Oracle Support Community.

Appendix C - How Long Should it Take to Run ORAchk

The time it takes to run the tool will vary based on the number of nodes in a cluster, CPU load, network latency, etc. Normally the entire process takes only a few minutes per node (i.e., less than 5 minutes per node). This is just a general guideline but if it takes substantially more time than this then there may be some other problem that needs to be investigated. Experience in the field has been that it normally takes about an hour for a 10 node cluster. It normally takes about 10 minutes for a two node system. In both cases those times include the tool environmental discovery phase and the entry of passwords. Adjust expectations accordingly for different sized systems or very busy systems.

Appendix D - Handling of Passwords by ORAchk

The tool does not store or save any passwords, either OS or database. For root passwords, the handling of passwords will depend on whether or not the EXPECT utility is installed

When expect is NOT present (default for all platforms except OEL5) the root password prompts are deferred and the user must closely monitor tool execution and enter the passwords interactively as prompted, once for each node of the cluster.

The tool leverages the EXPECT utility which is shipped by default as part of the OEL5 distribution for interactive password automation (see http://expect.sourceforge.org). Expect can be installed on other Linux distributions in order to automate interactive password handling.

When EXPECT is found the root passwords are "gathered" at the beginning of the processing and expect will supply them when needed at the root password prompts for each node so that the tool can continue without the need for further input from the user. The user will be asked if the root password is the same for all database servers of the cluster. If the user responds "yes" (the default) then the root password will be prompted for and validated once and that password will be used for all nodes of the cluster. If the user responds that the root password is not the same for all nodes of the cluster then the tool will prompt for and validate the root password for each individual node of the cluster.

Also, when EXPECT is found, in validating the root passwords the user will have three opportunities to type the correct password. If the user fails to type the correct password after three attempts the tool will move on to the next node and display a message stating that the password was still incorrect and that the checks dependent upon data collected from that node will be skipped. At that point the user can either cancel tool execution and get the problem resolved or continue but with the understanding that valuable data may be missing from the report. It is better to resolve the problem and start over than to be missing data

When EXPECT is utilized it is also possible that between the time that the root passwords are entered and validated and nodes for those passwords are reached that the passwords could have been changed. In that case a message will appear in the on-screen output stating that the password must have been changed and that the collections for that node will be skipped which means the checks for that node will also be skipped. The user can either allow the tool to continue to completion knowing that data and checks will be skipped or cancel tool execution (Control-C) and resolve the problem and try again. It is better to resolve the problem and start over than to be missing data.

In any case, any checks that are skipped (for any reason) will be logged and the reports listed at the end of tool execution will list any checks that were skipped and on which nodes.

“-debug” Option and Passwords

Because of the nature of debug operations within the bash shell, the “-debug” output file (and possibly others) will contain the strings that were entered in response to the prompts for operating system user passwords, and optionally the database login username and password when operating system authentication is not configured.

With 12.1.0.2.1, unless an ORAchk execution is abnormally terminated (kill -9, database server crash, etc), the ORAchk utility removes these strings by default.

If you do not wish to remove the password strings for some reason, use the following environment variable to preserve the original “-debug” output file and make a second file without the password strings:

```
$ export RAT_KEEP_ORIG_DEBUG_FILE=1
```

After using the RAT_KEEP_ORIG_DEBUG_FILE environment variable, you will see two files similar to these shown:

```
orachk_debug_20140918_103252.log
orachk_debug_20140918_103252.log.orig
```

The “.orig” file contains the password strings.

Note: If an ORAchk run using the “-debug” option was abnormally terminated, it is recommended to manually verify the directory in which ORAchk was launched and the timestamp output directory for files containing any password string entered in response to a prompt and edit the returned files to replace all occurrences of the password string discovered with a random string.

For example (password strings guest1 and tooeasy, ORAchk launched from /home/oracle/121021 /password_test):

```
$ egrep -lr 'guest1' *
orachk_randomadm01_fsdb_100714_074713/log/watchdog.log
$ egrep -lr 'tooeasy' *
$

egrep "guest1" orachk_randomadm01_fsdb_100714_074713/log/watchdog.log
send: sending "guest1\n" to { exp7 }
send: sending "guest1\n" to { exp6 }

$ sed -i s/guest1/random1/g orachk_randomadm01_fsdb_100714_074713/log/watchdog.log

$ egrep "guest1" orachk_randomadm01_fsdb_100714_074713/log/watchdog.log
$
```

Appendix E - Multiple Database Support

The ORAchk tool is designed for multiple database support.   The tool will present a list of running databases which are registered in the Grid Infrastructure. The user may choose one, all or enter a comma separated list of numbers which designate the databases listed. It is not necessary to stage the tool on multiple nodes in order to check database instances running on other nodes of the cluster.

All database logins are accomplished by the use of local bequeath connections and assume the existence of OS authentication in the database for the user executing the tool. In some configurations there could be multiple database homes all owned by the same OS user (e.g., oracle) or in others there could be any number of database homes all owned by different OS users. In the former case, run the tool as "oracle". In the latter case, stage and run the tool as the OS user which owns the RDBMS home which has the largest number of database instances running from that home as that user will not be able to login into database instances running from homes which it does not own. In order to scan the other databases the tool needs to be staged and run under each database home user account.

Appendix F - Uploading ORAchk Results and Patches to a Database for Reporting

For customers who have a large number of systems and databases it would be useful to upload the results of the audit checks done by the tool and/or the list of installed patches into database tables for use as a source of data for reporting.

Starting with ORAchk 2.2.4 Oracle now provides the ORAchk Collection Manager application. Collection Manager is Application Express (APEX) which provides an enterprise wide central repository for ALL ORAchk (and Exachk) collections. The APEX frontend provides an easy to use dashboard interface which users can track, trend, compare and report upon ORAchk findings. ORAchk Collection manager is the preferred approach to reporting on ORAchk collections. The Collection Manager application is bundled with ORAchk (CollectionManager_App.sql). For more information on ORAchk Collection Manager please see the Collection Manager Users Guide found in MOS Note: 1268927.2.

In order to support this feature a number of environment variables need to be set in the execution environment and three tables need to be created to receive the data, one for the audit check results, one for the patches installed on the systems and one to store the zipped archives that result from the collections:

AUDITCHECK_RESULT
AUDITCHECK_PATCH_RESULT
RCA13_DOCS

Reporting Table Specifications

Note: If you plan on deploying ORAchk Collection manager, the reporting schema documented below will be created when deploying Collection Manager in APEX.

Results table specification (e.g. auditcheck_result):

Version 2.2 added several additional columns to this table. Existing ORAchk users who are using the Database reporting features in ORAchk versions less than 2.2 MUST add these additional columns. The DDL to add this column is as follows:

```
SQL> alter table auditcheck_result add (
CHECK_ID VARCHAR2(40),
NEEDS_RUNNING VARCHAR2(100),
MODULES VARCHAR2(4000),
DATABASE_ROLE VARCHAR2(100),
CLUSTERWARE_VERSION VARCHAR2(100),
GLOBAL_NAME VARCHAR2(256));
```

Version 2.2.3 added 2 additional columns and constraints to the auditcheck_result table. Again, existing ORAchk users who are using the Database reporting features in 2.2 MUST add these additional columns and constrains when upgrading to ORAchk 2.2.3:

```
alter table auditcheck_result add (
     UPLOAD_COLLECTION_NAME VARCHAR2(256) NOT NULL ENABLE,
     AUDITCHECK_RESULT_ID VARCHAR2(256) DEFAULT sys_guid() NOT NULL ENABLE,
     COLLECTION_ID  VARCHAR2(40),
     CONSTRAINT "AUDITCHECK_RESULT_PK" PRIMARY KEY ("AUDITCHECK_RESULT_ID")
);
```

Version 12.1.0.2.1 added another 2 additional columns and constraints to the auditcheck_result table. Again, existing ORAchk users who are using the Database reporting features in 12.1.0.2.1 MUST add these additional columns and constrains when upgrading to ORAchk 12.1.0.2.1:

```
alter table auditcheck_result add (
     TARGET_TYPE    VARCHAR2(128),
     TARGET_VALUE   VARCHAR2(256)
);
```

Full DDL for the Results Table:

```
create table
auditcheck_result
(
     COLLECTION_DATE TIMESTAMP NOT NULL ENABLE,
     CHECK_NAME VARCHAR2(256),
     PARAM_NAME VARCHAR2(256),
     STATUS VARCHAR2(256),
     STATUS_MESSAGE VARCHAR2(256),
     ACTUAL_VALUE VARCHAR2(256),
     RECOMMENDED_VALUE VARCHAR2(256),
     COMPARISON_OPERATOR VARCHAR2(256),
     HOSTNAME VARCHAR2(256),
     INSTANCE_NAME VARCHAR2(256),
     CHECK_TYPE VARCHAR2(256),
     DB_PLATFORM VARCHAR2(256),
     OS_DISTRO VARCHAR2(256),
     OS_KERNEL VARCHAR2(256),
     OS_VERSION NUMBER,
     DB_VERSION VARCHAR2(256),
     CLUSTER_NAME VARCHAR2(256),
     DB_NAME VARCHAR2(256),
     ERROR_TEXT VARCHAR2(256),
     CHECK_ID VARCHAR2(40),
     NEEDS_RUNNING VARCHAR2(100),
     MODULES VARCHAR2(4000),
     DATABASE_ROLE VARCHAR2(100),
     CLUSTERWARE_VERSION VARCHAR2(100),
     GLOBAL_NAME VARCHAR2(256),
     UPLOAD_COLLECTION_NAME VARCHAR2(256) NOT NULL ENABLE,
     AUDITCHECK_RESULT_ID VARCHAR2(256) DEFAULT sys_guid() NOT NULL ENABLE,
     COLLECTION_ID  VARCHAR2(40),
     TARGET_TYPE    VARCHAR2(128),
     TARGET_VALUE   VARCHAR2(256),
     CONSTRAINT "AUDITCHECK_RESULT_PK" PRIMARY KEY ("AUDITCHECK_RESULT_ID")
);
```

Patch table specification (e.g. auditcheck_patch_result):
Version 12.1.0.2.3 added 1 additional the auditcheck_patch_result table.

```
alter table auditcheck_patch_result add (
     UPLOAD_COLLECTION_NAME VARCHAR2(256),
);
```

```
create table
  auditcheck_patch_result
       ( COLLECTION_DATE     TIMESTAMP(6) NOT NULL,
          HOSTNAME            VARCHAR2(256),
          ORACLE_HOME_TYPE    VARCHAR2(256),
          ORACLE_HOME_PATH    VARCHAR2(256),
          ORACLE_HOME_VERSION VARCHAR2(256),
          PATCH_NUMBER NUMBER,
         CLUSTER_NAME VARCHAR2(256),
         DESCRIPTION         VARCHAR2(256),
         PATCH_TYPE VARCHAR2(128),
         APPLIED NUMBER,
         UPLOAD_COLLECTION_NAME VARCHAR2(256),
         RECOMMENDED NUMBER
         );
```

DDL for the RCA13_DOCS Table:

```
CREATE TABLE RCA13_DOCS
  (DOC_ID NUMBER DEFAULT to_number(sys_guid(),''XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'') NOT NULL ENABLE,
   COLLECTION_ID VARCHAR2(40 BYTE),
   FILENAME VARCHAR2(1000 BYTE) NOT NULL ENABLE,
   FILE_MIMETYPE VARCHAR2(512 BYTE),
   FILE_CHARSET  VARCHAR2(512 BYTE),
   FILE_BLOB BLOB NOT NULL ENABLE,
   FILE_COMMENTS VARCHAR2(4000 BYTE),
   TAGS VARCHAR2(4000 BYTE),
   ATTR1 VARCHAR2(200 BYTE),
   UPLOADED_BY VARCHAR2(200 BYTE) DEFAULT USER,
   UPLOADED_ON TIMESTAMP (6) DEFAULT systimestamp,
   CONSTRAINT RCA13_DOCS_PK PRIMARY KEY (DOC_ID),
   CONSTRAINT RCA13_DOCS_UK1 UNIQUE (FILENAME));
```

Environment variables (with example settings):

```
export RAT_UPLOAD_CONNECT_STRING="(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = bonanza)(PORT =
1521)) (LOAD_BALANCE = yes) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = orcl)))"
export RAT_UPLOAD_TABLE=auditcheck_result
export RAT_PATCH_UPLOAD_TABLE=auditcheck_patch_
export RAT_ZIP_UPLOAD_TABLE=RCA13_DOCS
export RAT_UPLOAD_USER=auditcheck (schema owner of the table created for the prupose)
export RAT_UPLOAD_PASSWORD=auditcheck (password for the schema owner)
export RAT_UPLOAD_ORACLE_HOME=<path of database home> (optional, alternate home containing sqlplus that
you want to use for connecting in case not the current $ORACLE_HOME as derived by the tool from the
environment)
```

Note: Use the fully qualified address (as in the example above) for the connect string rather than an alias from the tnsnames.ora file so that it is not necessary to rely on tnsnames.ora file name resolution on all the servers where the tool might be run. The double quotes should be included.

When the first four above environment variables are set in the execution environment the tool will assume that the intent is to upload the data into the tables and at the end of the process it will attempt to upload the data. This process relies upon the environment being properly set, i.e. the connect string must be reachable, the username and password must be correct and the table name must be correct. If the tool is unable to connect to the database a message to that effect will be written to the log. If the RAT_UPLOAD_ORACLE_HOME variable is set the tool will invoke sqlplus from that home rather than attempting to invoke sqlplus from the current $ORACLE_HOME derived by the tool. If any of the first four environment variables are not set then the tool will not attempt to upload the data.

Appendix G - Optionally Excluding Audit Checks

To exclude an audit check before the html report is generated, construct a text file named "excluded_check_ids.txt" in the same directory where the ORAchk script and driver files have been installed.

The "excluded_check_ids.txt" contains the check identification numbers for the checks to be skipped, one per line. Each check is assigned a unique check identification number that does not change once assigned. Some checks will require one check identification number to be fully skipped, and some will require two. If a check has both an "COLLECTION_NAME" and an "AUDIT CHECK NAME" in the "exachk.log"file, it will require two entries in "excluded_check_ids.txt". If the check has only an "AUDIT CHECK NAME" in the "exachk.log"file, it will require only 1 entry in "excluded_check_ids.txt".

Determining Check IDs from an HTML Report

Beginning with version 2.2.3, there is a toggle button provided in the HTML report to make the check identification numbers visible or nor visible. The "Show Check Ids" link appears just below the "Table of Contents" section of the HTML report.

When the "Show Check Ids" link is clicked upon, the rows in the report tables expand to show the check identification numbers:

Check Id	**Status**	**Type**	**Message**	**Etc.**
DCDB11F4CA7C5701E04313C0E50AB5D3	FAIL	OS Check	vm.min_free_kbytes should be set as recommended.	Etc.

The "Show Check Ids" link also changes to "Hide Check Ids".

When the "Hide Check Ids" link is clicked upon, the rows in the report tables retract) to hide the check identification numbers:

Status	**Type**	**Message**	**Status On**	**Details**
FAIL	OS Check	vm.min_free_kbytes should be set as recommended.	All Database Servers	View

To exclude specific checks from being executed during the next ORAchk run, simply "Show Check Ids" and copy those Check Ids from the HTML report to the "excluded_check_ids.txt" within the directory in which the ORAchk "kit" was extracted to.

Determining Check IDs from Log Files

To find the check identification numbers to exclude, search for the name of the check you wish to exclude from the html report in the "orachk.log" file. This needs to be done using an "orachk.log" file from a previous complete ORAchk run. In version 2.2.2, you will find the “orachk.log” file in the “log” subdirectory created under the directory from the run. For example, if the run generated the subdirectory named “orachk_ratlnx01_mydb _052413_181326”, the “orachk.log” file will be at: “orachk_ratlnx01_mydb_052413_181326/log/orachk.log”.

To determine if a check can be excluded, and if it requires one or two check identification numbers, you can grep through the “orachk.log” file for the name of the check from the HTML report. Once you have found the check name, open the “orachk.log” file in an editor and search on the check name string. Each check has a block of text associated with it in the “orachk.log” file, usually followed by a line made of “-“. The check identification number will be above the dashed line. For certain checks there may be a COLLECTION_NAME as well as an AUDIT_CHECK_NAME.

To exclude specific checks from being executed during the next ORAchk run, simply copy the string after “CHECK_ID=” from the orachk.log file to the “excluded_check_ids.txt” within the directory in which the ORAchk “kit” was extracted to. For full exclusion of the check and its collected data you will want to exclude both the COLLECTION_NAME and AUDIT_CHECK_NAME check ids (where applicable).

Excluding Checks Based on a Profile

Users can exclude checks based on a profile using a new option introduced in orachk 2.2.4 called excludeprofile. Checks from the profile mentioned after excludeprofile will not be executed when the tool is run and the excluded profile will be listed in the report header.

Removing Checks from an Existing HTML Report

Starting with ORAchk 2.2.3, you now have the ability to remove checks from an existing HTML report. To do so you will click the “Remove finding from report” link just below the Table of Contents. This will display an X next to each audit check within the HTML report. To remove a given check simply click the X. Once the desired checks have been removed you may click the “Hide remove Buttons” link, again just below the Table of Contents, to display the revised report.

Note: Removing the checks using the above method does NOT actually modify the HTML report. To backout changes you may simply reload the report in your browser. If you wish to make the changes permanent in the report you MUST use the browser’s “save page” functionality.

Note: When uploading results to the ORAchk Collection Manager application mentioned earlier it is also supported to exclude (ignore) findings within the application based on business unit, system or individual collection. See the ORAchk Collection Manager Installation and User Guide for more details but this feature provides flexibility for excluding or ignoring findings for checks.

Appendix H - Special Notes on USERIDs and PASSWORDS

As a means of providing higher security when using the tool, passwords are not stored. The tool uses operating system or standard Oracle username and password authentication mechanisms.

If the user executing the tool has an account which is set up for OS authentication in Oracle then no username and password will be required for database log in. However, if the user is not set up with OS authentication for database access then the user will be prompted with the standard Oracle database authentication prompts.

Appendix I - Special Notes on Output Directory

To limit security vulnerabilities, the permissions of the tool output directory should be set as restrictive as possible. The output directory could contain sensitive configuration information and, when no other mechanism is available, temporary data collection files.

Appendix J – Output File Maintenance

Whenever ORAchk is run it creates a subdirectory using a naming convention that begins with "orachk" and includes a date and time (e.g., orachk_SIEBXL_072611_141001) and a .zip archive that contains the contents of the subdirectory (e.g., orachk_SIEBXL_072611_141001.zip) at the same level on the filesystem as the tool itself. The total size of the subdirectory and .zip file is expected to be somewhat less than about 5M on the filesystem. The exact size may vary depending upon how many nodes and how many databases there are in the system. While it is expected that orachk will be run at times as described in the "When To Run ORAchk" section over time the number of files will build up and the customer may want to perform maintenance and clean out older files and/or subdirectories.

Each customer may have different requirements and therefore it is incumbent upon the customer to define processes and procedures for implementing their own strategy for maintaining these subdirectories and files. Some customers may choose to archive the files to a different system. Some customers may want to delete the subdirectories and keep the .zip files. Still other customers might opt to delete both the older subdirectories and .zip files.

Oracle recommends that the customer implement cron jobs or other scheduling tools (e.g., Oracle Enterprise Manager) to implement their own process and procedure for this task based on their individual requirements.

Appendix K – Maximum Availability Architecture Scorecard (MAA)

The MAA Scorecard is a set of Maximum Availability Architecture Best Practices and the findings related to them. The purpose is to give the customer a Scorecard for their preparedness for various kinds of faults that can occur in an Oracle database environment. The MAA Scorecard is now presented by default in orachk because MAA is considered to be a very important concept and set of features. However, some customers may not have implemented Oracle Data Guard and it is most helpful when the customer has implemented Oracle Data Guard standby databases but even Oracle data guard is not implemented, its recommended to run orachk with MAA scorecard. If it is not desirable to include the MAA Scorecard in the analysis it can be suppressed by running orachk using the -m argument, e.g.

```
$ ./orachk –m
```

Appendix L – Upgrade Readiness Checks for 11.2.0.3 and higher

There are numerous documents which discuss various aspects of upgrading to Oracle RAC version 11.2.0.3- Upgrade Companion, Upgrade Guide, Oracle Database Documentation, Manual Upgrade Checklist, to name a few. The upgrade readiness module of the tool automates the checking of many of those things such as best practices, patch pre-requisites, configuration pre-requisites, etc. The goal is to simplify and enhance the reliability of the upgrade experience. While we have attempted to cover as many points as possible it is still advised that customers review the upgrade documents as some concepts or individual items do not lend themselves easily to automation. See the usage section for the syntax for running upgrade readiness checks.

Appendix M – Configuring SUDO Access for ORAchk

In order to take advantage of the full functionality of ORAchk, root access is required to run the root specific checks. To facilitate these checks in silent mode we MUST configure passwordless SUDO and utilize the -s (more on this later) option to execute ORAchk. Configuring SUDO will also allow for parallel ORAchk execution (new in 2.2.1) without having the EXPECT utility installed on the system.sudo configuration is not required if orachk deamon is configured with or without auto restarting daemon feature using init. To configure SUDO add the following line to the sudoers file on each of the cluster nodes using visudo command replacing "<oracle>" with the appropriate RDBMS software owner:

```

<oracle> ALL=(root)  NOPASSWD:$HOME/root_orachk.sh
```

Appendix N – Platform-wise Checks that Require Root Access

For a complete list of Checks for each platform that require root access please refer to the ORAchkPlatform-wiseChecksthatRequireRoot.pdf attached to Note: 1268927.1.

Appendix M – Upgrading ORAchk

New versions of ORAchk will be released periodically. Frequently check MOS Note: 1268927.1 for new versions and/or subscribe to the ORAchk MOS community to keep up to date.

To Upgrade ORAchk from version 2.2.5:

Note: Versions prior to 2.2.5 require manual upgrade (rename the old directory and create new directory)

1. Download the new version of ORAchk from MOS Note: 1268927.1 to a shared network staging

directory (do not unzip)
2. Set the environment variable $RAT_UPGRADE_LOC to the staging directory
3. Execute “orachk –upgrade” to upgrade ORAchk

To Upgrade ORAchk from version greater than or equal to12.1.0.2.1:

With ORAchk 12.1.0.2.1, by default, when ORAchk starts it will look for a new version in $ORACLE_HOME/suptools or $RAT_UPGRADE_LOC if specified. If a new version exists ORAchk will automatically be upgraded to the new version. Best practice is to stage orachk.zip on a network share accessible to ALL environments running ORAchk and set the environment variable $RAT_UPGRADE_LOC to that location.

Note: To bypass auto-upgrade, you may execute “orachk –noupgrade”. If the version of ORAchk is older than 120 days, the –noupgrade option will NOT be effective.

1. Download the new version of ORAchk from MOS Note: 1268927.1 to a shared network staging directory (do not unzip)
2. Set the environment variable $RAT_UPGRADE_LOC to the staging directory
3. Execute ORAchk, upon execution ORAchk will prompt to confirm the upgrade

Appendix O – Windows Cygwin Requirement

Cygwin is essentially a utility that offers a Linux-like environment on a Microsoft Windows host. Technically, it is a DLL (cygwin1.dll) that acts as a Linux API layer providing substantial Linux API functionality.

In order to make it possible for customers to use ORAchk in the Windows environment Oracle recommends installing Cygwin on the database servers. Cygwin provides a bash scripting shell environment that is compatible with ORAchk. Once you install Cygwin, you can configure the SSH Daemon on the host and SSH Daemon only needed for Oracle RAC installations.

For complete instructions on how to install Cygwin for use by ORAchk, see the “Supported Environments” tab within MOS Note: 1268927.1 and drill down on Windows Support.

What's New By Version

Version 12.1.0.2.3
- Linux on System Z (RHEL 6, RHEL 7, Suse 12)
- Oracle Enterprise Linux 7
- Single Instance ASM Checks
- Upgrade Validation checks for 12.1.0.2
- Enhanced Golden Gate Checks
- Enterprise Manager Agent Checks
- Enhanced Reporting to highlight checks that ORAchk cannot gain full visibility for (checks that require manual validation)

- Improved health Score Calculation (you can now hit 100%)
- Over 50 new health checks
- Bug Fixes

Version 12.1.0.2.1

- Windows Support (Cygwin Required, see Appendix O for Cygwin Requirement)
- Support for execution as “root” to simplify execution in role separated environments
- Extend baseline comparisons to collections
- Ability to crate user defined collection names using tags
- Self Upgrade from a single shared location
- ASM Checks enabled for Single Instance environments
- New checks and bug fixes

Version 2.2.5

- Execution of checks in parallel on multiple databases
- Allowing multiple runs through ORAchk daemon
- Move away from /tmp as temporary destination, $HOME directory is the new temporary destination (allowed to be overridden by setting RAT_TMPDIR)
- Enhancements in health check score calculation to include skipped checks
- Enhanced reporting for cluster wide checks
- ORAchk upgrades now handled via the new "-upgrade" option
- Granular help at the command line option level, for example: ./orachk –profile –h
- New Profile for pre-install checks
- Best Practices for single instance pre/post upgrade (Upgrade Readiness)
- Collection Manager is now bundled with ORAchk (CollectionManager_App.sql)
- New checks and bug fixes, including
    - Enterprise Cloud control Best Practices for the database tier
    - Over 40 new Health Checks for the Oracle Stack
    - Expands to additional product areas in E-Business Suite Workflow & Oracle Purchasing

Version 2.2.4

- ORAchk daemon auto-start mode after node reboot (init integration)
- Merge multiple ORAchk collection reports
- Exclude checks based on profile
- Upload of installed patches to database
- Collection Manager for ORAchk, RACcheck and Exachk
- ORAchk signature file in /tmp on all nodes to verify last ORAchk run
- New checks and bug fixes, including
    - 30 Oracle Ebusiness AP module data integrity checks
    - 12 new Database checks
    - 8 new Solaris system checks

Version 2.2.3

- GoldenGate Best Practice checks (applicable only for databases running GoldenGate)
- Database Consolidation Best Practices in MAA Score Card
  ORAchk Daemon enhancement allowing for execution at specific scheduled dates/times in

addition to interval option

- ORAchk Daemon diff check of current and current-1 reports and emails if differences are found
- Ability to exclude checks based on the check name
- Excluded checks are listed in html report
- Visual progress indicators have been added to confirm script progression at key points
- New checks and bug fixes

Version 2.2.2

- Execution as the root user for sysadmin profile is now possible, e.g. ./orachk -profile sysadmin
- ORAchk daemon feature to automate execution of ORAchk non-interactively at predefined interval
- Support for Solaris Sparc 11
- Upgrade best practices for 11.2.0.3, 11.2.0.4 (not yet released) and 12c (not yet released)
- ORAchk output directory restructure
- Standard health check is now included in post upgrade
- New checks and bug fixes

Version 2.2.1

- Execution performed in parallel on all nodes (for root checks, OS expect utility or SUDO is REQUIRED to enable this functionality)
- Use of profiles to execute a subset of checks, e.g. DBA, Sysadmin, ASM
- Ability to compare two ORAchk
- New checks and bug fixes

Version 2.2.0

- Single Instance Database Support (including Oracle Restart)
- High Availability checks
- RAC One Node Support
- Additional Columns added to Database Reporting tables for increased flexibility (See Appendix F)
- New checks and bug fixes

Version 2.1.6

- New Supported Platforms:
    - OEL and RHEL 6
    - HP-UX (BASH shell 3.2 or higher required)
    - AIX 7 (BASH shell 3.2 or higher required)
- HTML Report ADA Compliance
- New checks and bug fixes

Version 2.1.5

- Upgrade Readiness checks (11.2.0.3 target version and above)
- MAA Scorecard is now displayed by default (use -m argument to suppress)
- Bug fixes and other general improvements

Version 2.1.4

- Remote database support. orachk will check database best practices for database instances running on nodes other than the tool execution node
- Solaris x86-64 support
- Enhancements to improve multiple database version support
- New Checks and bug fixes

Version 2.1.3

- 11.2.0.3 support
- Suse 11 support
- MAA Scorecard
- Bug fixes

Version 2.1.2

- New HTML based report
- Bug fixes