orasetup User Guide

Overview

orasetup is a korn shell/bash shell script that is a replacement for Oracle's "oraenv" script. It is meant to be used to set up an Oracle environment for database or application server management. orasetup sets all of the environment variables, as specified in the Oracle installation guides, for a given release and port. orasetup also supports a unique ability to handle additional Oracle environments and utilities that oraenv does not. These include things like the Oracle Network listener, Oracle Names, the OEM Intelligent Agent and dbconsole and the various application server environments. The following documentation will detail the use and operation of the orasetup utility.

Source

orasetup is available as an open-source utility and is part of the Oracle Management Tool Suite. orasetup can be downloaded from from Github at https://github.com/AndyRivenes/orautil.

Installation

orasetup is a korn/bash script so it is just a text file. Once downloaded from Github the installation of orasetup involves simply locating it in a directory in the shell's PATH, making it executable and updating the ORACLE_BASE= location in the script.

The recommended approach is to place orasetup in a "script" directory as in $ORACLE_BASE/local/script. This is meant to be OFA compliant and the $ORACLE_BASE/local directory should be fairly standard on most systems. Another alternative is to locate it in the local bin directory (i.e. /home/oracle/bin). The recommended permissions are 755 since it is meant to be used by any user.

In the DEPVAR function towards the beginning of the script there is a setting for the ORACLE_BASE directory. This should be changed to match the actual ORACLE_BASE location. The following is what the code looks like:

```
#
# Oracle Environment specific variables
#
if [ "$ORACLE_BASE" = ""
then
  ORACLE_BASE="/u01/app/oracle"
  export ORACLE_BASE
fi
```

The script makes use of the following UNIX utilities, and they should be in the PATH:

- sed
- awk
- uname
- grep

The following installation steps are documented in the orasetup banner:

```
1) Recommend locating orasetup in the $ORACLE_BASE/local/script
   directory.
2) Recommend adding $ORACLE_BASE/local/script to the oracle
   account's PATH.
3) Verify that all databases are set correctly in the oratab file.
4) Modify the DEPVAR function to set environment specific variables.
   The only required modification is to the ORACLE_BASE setting.
   NOTE: If using Application Server services then the LVAR setting
   for JAVA_HOME must be set.
5) If using the bash shell then uncomment the bash interpreter call
   at the top of this file and move up to the first line (e.g. it
   should look like: "#!/bin/bash line", and comment out or remove
   the "#!/bin/ksh" line).
```

Configuration

As with the oraenv script that Oracle supplies, and along with many other Oracle utilities, orasetup uses the file *oratab* to determine how to set an environment. Specifically, the format of the oratab file is:

```
<db sid>:<ORACLE_HOME>:<Y|N|W>
```

along with comments starting with a "#".

orasetup interprets the oratab file exactly the same as the Oracle utilities with the exception that an "N" flag in the third field causes orasetup to set the TWO_TASK environment variable instead of the ORACLE_SID environment variable. This used to be known as a "server partitioned" environment and is still useful when the client tools are either remote to the database or are located in a different ORACLE_HOME (if on the same machine). When setting the SID field in the oratab file for a server partitioned environment, the value should match the Oracle Network alias for the target database.

Oracle allows a "*" to be specified instead of an actual SID in the first column of the oratab file. In recent versions the installer will add this entry if no database is created at the time the ORACLE_HOME software is installed. If orasetup is invoked without a SID parameter, it will set the Oracle environment based on the ORACLE_HOME supplied in the "*" entry if it exists.

oratab Options

orasetup supports extensions to the oratab file through the use of commented labels to identify additional Oracle utilities. This allows orasetup to support environment setup of virtually any Oracle utility. Some labels support additional optional parameters as well. An example is the SNET label which supports an optional parameter for the listener name. This allows orasetup to support multiple listeners whether they are run from the same or different ORACLE_HOMEs.

The following comments are recognized by orasetup in the oratab file:

```
#NETV1 - SQL*Net V1
#SNET  - SQL*Net V2/Net8
         option: Listener name
         Note: multiple entries are supported
#NAMES - Oracle Names
#AGENT - Oracle Intelligent Agent, not used for the 10g
         dbconsole.
#OBACK - Obackup (EBU prior to 2.2)
#EBU   - EBU 2.2
#OID   - Oracle Internet Directory
#OEM   - OEM Managment Server
         option: 10g database SID, supports 10g dbconsole
         Note: multiple entries are supported
#OWS3  - Oracle Web Application Server 3
#OC4J  - Oracle Containers For J2EE (standalone)
#AS9   - Oracle9i Application Server
         option: Infrastructure database alias
#AS10  - Oracle 10g Application Server
         option: Infrastructure database alias
#GRID  - Oracle Grid Control, see AS10
#HTTP  - Oracle HTTP Server
#OCA   - Oracle Certificate Authority
#CRS   - Oracle Cluster Ready Services
```

Note: Just the comment and label are included in the oratab file (i.e. #SNET), the description provided here only serves as a reference to the label.

Usage

orasetup must be invoked with a leading ".". This is equivalent to "sourcing" a script in the C shell and tells the korn or bash shell to run the script in the current environment. This is the only way to preserve the changes a shell script makes to the user's environment. This behavior is the same for oraenv and any korn or bash shell script.

orasetup accepts zero, one or two arguments, depending on the oratab setup. The following is the orasetup syntax:

```
orasetup [ SID | <utility> [ alias ]
           unset | opatch | ssh
```

```
          help | ? | version| menu ]
  Options:
    SID – Database SID based on first label in the oratab file.
    <utility> [ alias ] – recognized utility and optional alias
      as defined by #<utility>: format in the oratab file
    unset    – Unset all variables set by orasetup
    opatch   – Add $ORACLE_HOME/OPatch to the PATH
    ssh      - Runs ssh.env in $HOME/.ssh (default) to set
      ssh equivalence
    help | ? – Display this help message
    version  - Display orasetup version
    menu     - Display database and known utilities defined in
      the oratab file
```

Return Codes

The following return codes are used by orasetup. This facilitates the usage of orasetup in other scripts and is called by several other utilities in the Oracle Management Tool Suite.

```
Return codes:
   0 - Normal
   1 - Error condition
   2 - TWO_TASK has been set
   3 - Warning, 32-bit mode required
```

Option Details

Environment Variable Setting

orasetup will set the following environment variables:

- ORACLE_BASE
- ORACLE_HOME
- ORACLE_SID or TWO_TASK
- PATH
- LD_LIBRARY_PATH or SHLIB_PATH
- NLS_LANG (optional)
- ORA_NLS or ORA_NLS32 or ORA_NLS33
- ORACLE_TERM
- EPC_DISABLED

It should be noted that orasetup will always unset all Oracle related environment variables first. It does this to try and ensure a "clean" environment before starting to set any environment variables. orasetup will first cycle through all of the oratab entries and search for any occurrences that match ORACLE_HOME or ORACLE_SID/TWO_TASK settings.

Local Variable Support

Additional local variables are also supported through the use of special "LVAR" settings in the DEPVAR function. The format is LVAR[#] where # is a sequentially increasing number starting with 1. By default, the LVAR1 variable is set to "$ORACLE_BASE/local/sql". Other commented examples include NLS_LANG, TNS_ADMIN, LD_ASSUME_KERNEL and other useful settings. These simply need to be uncommented and follow the sequentially increasing numbering scheme. Any other values can be added as well.

Local Script Support

There is also the ability to run a SID specific script of the format:

$GBL_USERENV/${ENVFIL}.env

where the $GBL_USERENV variable is set in the DEPVAR function to the path where the $ENVFIL.env script is located. The value of the $ENVFIL is expected to be either the ORACLE_SID or TWO_TASK if set.

Additional Options

Two other options have been built into orasetup to support the Oracle Management Tool Suite bkctrl utility and to enable the automatic running of the Oracle Applications 11i environment setup script.

- If orasetup finds a bkctrl config file for the SID being set up it will run it (the file is run with a leading "." so all changes are permanent for the current shell). Typically, this file is found in the $ORACLE_BASE/local/bkup directory with the name <sid>.config where <sid> matches the database SID orasetup found in it's first argument.
- If orasetup finds a <sid>.env file in the $ORACLE_HOME directory, then it will run it (the file is run with a leading "." so all changes are permanent for the current shell). This is useful for Oracle Applications 11i environments to set utility paths. This is consistent with how Rapid Install configures the environment file setup for the database tier.

Additional Options

Application Server Support on Linux 64-bit Platforms

Currently all of the 10g Application Server and prior releases only function as 32-bit applications. On 64-bit Linux platforms a special 32-bit emulation shell must be used to run the Application Server software. orasetup will check for this and if not running in a 32-bit emulation shell for the various 10g AS options, including Grid Control, it will display a message and return a code of 3.

Standby Database Support

Standby database support has been added to oratab to match the usage of standby database support in the rest of the Oracle Management Tool Suite. The idea is that standby databases need special treatment so that they are not accidently opened or damaged. As such, the Oracle Management Tool Suite guidelines set a standby database's third flag entry in the oratab file to a N and then add a separate comment to identify the database as a standby database. This has the effect of causing orasetup to always set the TWO_TASK environment variable. The idea being that the user must consciously switch to the correct ORACLE_SID environment variable manually in order to connect to the database. The hope is that this will act as a safety check to help protect the standby database. The other tools in the Oracle Management Tool Suite recognize these oratab entries and treat the database as a standby database as well. The following are example oratab entries for a standby database:

```
DBSID:/u01/app/oracle/product/10.2.0/db_1:N
#STANDBY:DBSID:MANAGED:DBSID_dg2
```

OPatch Environment Setup

When working with the OPatch utility it is handy to add the OPatch directory to the PATH so that opatch can be invoked without fully qualifying the path. orasetup supports this by adding an “opatch” option that can be run once the database environment is set. This feature assumes the site has kept the OPatch directory located in $ORACLE_HOME, which is where Oracle initially places it for distributions that have it installed.

Unsetting the Oracle Environment

The orasetup "unset" code can be invoked directly with the UNSET parameter. This can be useful when it is desired to have no Oracle related environment variables set.

SSH Equivalence

orasetup supports setting SSH equivalence for RAC environments with a “ssh” option. When invoked with “ssh” orasetup will search for and run a “ssh.env” file in the directory defined by the global variable $GBL_SSH (defaults to $HOME/.ssh).

Menu

A “menu” option has been added to orasetup that will parse the oratab file and display the defined databases and known utilities. It is helpful to set this option at the end of the .profile file to display when logging in:

```
#
# Invoke orasetup menu function
#
. orasetup menu
```

Examples

Normal Setup Routine

```
/u01/app/oracle$ . orasetup DBSID
/u01/app/oracle$ echo $ORACLE_SID
DBSID
/u01/app/oracle$
```

All other Oracle related environment variables for this release have also been set.

TWO_TASK Setup

```
/u01/app/oracle$ cat /etc/oratab
DBSID:/u01/app/oracle/product/10.2em/db10g:Y
REMSID:/u01/app/oracle/product/10.2em/db10g:N
#GRID:/u01/app/oracle/product/10.2em/oms10g:DBSID
#AGENT:/u01/app/oracle/product/10.2em/agent10g
#SNET:/u01/app/oracle/product/10.2em/db10g:LISTENER
/u01/app/oracle$ . orasetup DBSID
/u01/app/oracle$ echo $ORACLE_SID
DBSID
/u01/app/oracle$ . orasetup REMSID
/u01/app/oracle$ echo $ORACLE_SID

/u01/app/oracle$ echo $TWO_TASK
REMSID
/u01/app/oracle$
```

The preceding shows how orasetup treats the Y or N flag in the orasetup file. For an entry with a Y flag the environment will be setup and the ORACLE_SID environment variable will be set with the first parameter (i.e., the Oracle SID). In the second case the third flag is a N and now orasetup will set the TWO_TASK environment variable.

Setup Details

This setup shows the full routine for an environment that might not start out with korn shell, but that has a .profile to set the $ORACLE_BASE/local/script.

```
$ ksh
$ . ./.profile
/u01/app/oracle$ echo $PATH
/u01/app/oracle/bin:/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:.:/u01
/app/oracle/local/script
/u01/app/oracle$ . orasetup DBSID
/u01/app/oracle$ env
_=/bin/env
PATH=/u01/app/oracle/bin:/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:.
:/u01/app/oracle/local/script:/u01/app/oracle/product/10.2em/db10g/bin
ORACLE_BASE=/u01/app/oracle
SQLPATH=/u01/app/oracle/local/sql
ORACLE_HOME=/u01/app/oracle/product/10.2em/db10g
ORACLE_SID=DBSID
PS1=/u01/app/oracle$
ORA_NLS10=/u01/app/oracle/product/10.2em/db10g/nls/data
LD_LIBRARY_PATH=/usr/dt/lib:/usr/lib:/usr/lib/X11:/u01/app/oracle/produ
ct/10.2em/db10g/lib:/u01/app/oracle/product/10.2em/db10g/ctx/lib
EPC_DISABLED=TRUE
ORACLE_TERM=vt220
BKUPID=DBSID
```

First, we invoke korn shell. Then we "source" the .profile and show that the local/script directory has been added to the PATH. Now we can "source" orasetup directly and then an "env" is run. Only the orasetup affected variables are shown.

Unsetting Variables

From the previous example's environment, the following shows the result of the "unset" option.

```
/u01/app/oracle$ . orasetup unset
/u01/app/oracle$ env
_=/bin/env
PATH=/u01/app/oracle/bin:/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:.
:/u01/app/oracle/local/script
ORACLE_BASE=/u01/app/oracle
/u01/app/oracle$
```

Now only the ORACLE_BASE environment variable remains, and all Oracle related directories have been removed from the PATH environment variable.

OPatch Setup

The opatch option can be run after the database environment is set up to add the $ORACLE_HOME/OPatch directory to the PATH environment variable. This makes it much easier to use the opatch utility.

```
/u01/app/oracle$ . orasetup DBSID
/u01/app/oracle$ . orasetup opatch
/u01/app/oracle$ opatch version
Invoking OPatch 10.2.0.3.0

OPatch Version: 10.2.0.3.0

OPatch succeeded.
/u01/app/oracle$
```

10g AS Setup in a 64-bit Linux Environment

orasetup will automatically detect a 64-bit environment for the GRID, AS10 and AS9 options and will set 32-bit emulation mode with the “linux32” command.

```
/u01/app/oracle$ . orasetup AS10
Warning - Running 64-bit Linux
AS10, GRID and HTTP options require 32-bit emulation mode
Set with the linux32 command
/u01/app/oracle$ linux32 ksh
/u01/app/oracle$ . ./.profile
/u01/app/oracle$ . orasetup AS10
/u01/app/oracle$ uname -a
Linux nspws-3.llnl.gov 2.6.9-55.0.2.ELsmp #1 SMP Tue Jun 12 17:58:20
EDT 2007 i686 athlon i386 GNU/Linux
/u01/app/oracle$
```