Top Tip: Some thing old, something new(-ish)

By Tim Onions TOdC Limited

Like many seasoned Oracle professional (a term I like to use about myself – professional that is not the seasoned part) I have a custom login script for SQL*Plus to set-up my environment, configure NLS settings and change the SQL> prompt - so I can see at a glance what database I am working on. However, in recent weeks I found that the information I was gathering had one important item missing – it did not tell me which host server the database resided on. As the work I am currently involved in uses a lot of external files, directories and the like it requires frequent switching from PC client development environment tools to a unix shell on the database server. To add to the mix a rigorous, thorough and seemingly ceaseless exercise in server moves, rationalizations and virualisations it was becoming tedious to constantly have to interrogate the database to find out where to pitch my putty.

Coming from a DBA background my first instinct was to select HOST_NAME from V$INSTANCE – simple enough and after a few times even my desperate typing could get it right 9 times out of 10 and in, what I felt, a pretty impressive speed.

```
SQL> SELECT host_name
FROM v$instance;

HOST_NAME
------------------------------------
todclnxdb01
```

Pride goes before a fall, as they say (who are ***they*** exactly anyway?). A recent move from DBA work to design and development work scuppered this – developers do not get access to V$ tables by default and have to put up a pretty strong case (and wade through levels of red-tape) to get access even to the most benign such data. Having been through the DBA end of SOX, any indignation at not letting little ole ME have access to a V$ table was short lived and so rather than fight for the right to party with V$ tables I went hunting for something more universal.

When I stumbled across the user environment variable *SERVER_HOST* I thought I had cracked it. A simple SYS_CONTEXT call will get the name of the host of the database you are connected to e.g.:

```
SQL> SELECT SYS_CONTEXT('USERENV','SERVER_HOST')
FROM dual;

SYS_CONTEXT('USERENV','SERVER_HOST')
------------------------------------
todclnxdb01
```

Beautiful, simple and consistent with my existing login.sql code to get database name (via SYS_CONTEXT('USERENV', 'DB_NAME') or SYS_CONTEXT('USERENV', 'INSTANCE_NAME') if using RAC). However, my pride was about to get a second fall. Yes, it works wonderfully – but only from 10g and beyond. The “SERVER_HOST” context was not introduced until then.

Then, out of the ark came an unexpected answer. Since Oracle7 when webDB/HTML DB existed (or whatever it was called in those days) there has been a set of utility packages used, primarily, for helping the programmer do work around the dynamic generation web pages [Aside: anybody remember owa_pattern – the much under-used precursor to REGEX built-ins in the database?]. One of these has a call to get the host server name - utl_inaddr.get_host_name – and it is still there in 10g. So, to get the host server name of the database instance you are currently connected to simply use the following (or similar) query:

```
SQL> SELECT utl_inaddr.get_host_name
```

```
FROM dual;

GET_HOST_NAME
------------------------------------
todclnxdb01
```

It was not necessary for me to have this shown in the SQL> prompt (which would have become a bit cluttered as a result in any case) so my LOGIN.SQL simply selects the server host name into a SQL*Plus variable and anytime I need to see the name of the host server of the database I am connected to I just type DEF (even those three letters I can get in the right order most of the time).

Quick UNIX/LINUX bonus tip

For those of you who are not already shell scripting experts - ever wanted to turn on spooled output form a UNIX shell script ubt from within the script as opposed to kicking off the script with output redirected via the >/*filename* pipe? Easily done with the EXEC command. e.g.

```
exec > /logs/myshellscript.log
exec 2>&1
…
```

Add this to the start of your script and you never have to kick it off with redirection again if you always want a log file. Add this to a script run by your application but which you have no means to access/change whatever it is that starts the script in the first place and you will be able to gather "debug" information.

Want to know what each line was doing in the script as part of the output – set debug mode (via set –x) and change the prompt seen in debug mode (PS4) to '$0.$LINENO+ ' so that your spooled output includes the script name and script line number that each line of output came from. So add the following to your script, at the start:

```
export PS4='$0.$LINENO+ '
exec > /tmp/script.log
exec 2>&1
set –x
..
```

About the Author

Tim Onions is an independent database consultant with over 15 years' experience with Oracle databases. Tim specialises in the application and database design of high performance systems, as well as tuning and optimisation techniques and can be reached at Tim.Onions@TOdC.co.uk