**CHẠY SCRIPT LẤY THÔNG TIN TRÊN MÔI TRƯỜNG THỬ NGHIỆM**

***Thông tin môi trường thử nghiệm****:*

|  |  |
| --- | --- |
| **OS platform** | Red Hat Enterprise Linux Server release 6.5 (Santiago) x64 |
| **Oracle DB version** | 11.2.0.4.0 (x64) |
| **Oracle SID** | EBANKDC2 |

1. **Sử dụng RDA thống kê OS, cluster, DB**

* Check perl, nếu đã được install thì sử dụng rda.pl còn không thì dung rda.sh

|  |
| --- |
| $ perl –V |

* Config rda, bước này sẽ lựa chọn các thông tin cần thống kê

|  |
| --- |
| $ cd rda $ chmod +x rda.sh  Or  $ ./rda.sh –S |

Xem log: “**run\_rda\_1.txt**”

* Chạy rda

|  |
| --- |
| $ ./rda.sh |

Xem log: “**run\_rda\_2.txt**”

Kết quả: “**RDA\_output\_custdb2.zip**” (chạy RDA\_\_start.html để xem)

1. **Sử dụng script lấy thống kê AWR**

* Lấy thông tin AWR dạng html

|  |
| --- |
| $ sqlplus / as sysdba @rpthtml\_genall.sql |

Xem log: “**run\_awr\_html.txt**”

Kết quả: “**AWR\_ebankDC\_06\_AUG\_2015\_00.zip**”

* Lấy thông tin AWR dạng text

|  |
| --- |
| $ sqlplus / as sysdba @*rpthtml*/get\_snap\_awr.sql |

Xem log: “**run\_awr\_txt.txt**”

Kết quả: “**tmp\_odp.zip**”

1. **Sử dụng script check DB**

* Tạo bảng chứa schema sử dụng lưu dữ liệu

|  |
| --- |
| SQL> CREATE TABLE APP\_USERS as SELECT USERNAME FROM DBA\_USERS WHERE USERNAME in ('SCOTT');  SQL> @healthcheck.sql |

Xem log: “**run\_healthcheck.txt**”

Kết quả: “**pfcheck\_06082015\_00.58.xls**”

1. **Các đoạn script đính kèm**

|  |  |
| --- | --- |
| RPTHTML\_GENALL.SQL | prompt -------------------------------------------  accept numdays prompt "How many days AWR you want to get?..."  define numdays='&&numdays';  set echo off  set termout on  prompt "Generating AWR Reports..................................."  @rpthtml/awr\_gen\_batch\_days.sql  undefine dbname  undefine timestamp  undefine suffix  undefine numdays  undefine days  set termout on  prompt "Reporting was succesful, checking all zip files........."  prompt  exit |
| GET\_SNAP\_AWR.SQL | -- DISCLAIMER  -- ==========  -- This script come without warranty of any kind.  -- Use them at your own risk. ONDATAFINE assumes no liability for any sort of damages using these scripts may cause.  --  -- USAGE  -- =====  -- You must have writting permissions on the current directory and enough space  -- Connect as sqlplus / as sysdba  -- run this script @get\_snap\_awr  -- The zip commande is in ORACLE\_HOME/bin in 10g else you have to install it  -- This script is platform independent.  connect / as sysdba;  set define ~  set serveroutput on  set ver off  set feed off  set linesize 200  host rmdir /Q /S tmp\_odp  host rm -fr tmp\_odp  WHENEVER SQLERROR EXIT 1  WHENEVER OSERROR EXIT 2  PROMPT Create or check of temporary directory : tmp\_odp  host mkdir tmp\_odp  -- Check writting permissions on temporary directory  spool tmp\_odp/test\_odp.log  spool off  PROMPT Temporary directory is ok  alter session set nls\_date\_language ='AMERICAN' ;  prompt  prompt IMPORTANT: Use format DDMMYY for the variable 'v\_date'  undefine v\_date  define v\_date=~v\_date  spool tmp\_odp/tmp\_awrondataperf.lst  declare  min\_snap number;  snap\_id2 number;  snap\_bidon number;  min\_snap\_id varchar2(30);  startupdb varchar2(30);  instance varchar2(30);  filereport varchar2(100);  filereport2 varchar2(100);  filereportzip varchar2(100);  portstring varchar2(100);  cursor c1 is  select  A.snap\_id-1 v\_snap\_id  from  sys.wrm$\_snapshot A, v$database B  where  A.begin\_interval\_time between to\_date('~v\_date 23:59:59','DDMMYY HH24:MI:SS')-1  and to\_date('~v\_date 23:59:59','DDMMYY HH24:MI:SS')  and A.dbid=B.dbid  order by  1;  c1\_rec c1%rowtype;  first\_loop number := 0 ;  begin  dbms\_output.enable(1000000);  filereport2 := 'odp'||'\_'|| '~v\_date' || '\_\*.txt' ;  select dbms\_utility.port\_string into portstring from dual ;  select substr(instance\_name,1,8) into instance from v$instance ;  for c1\_rec in c1 loop  if (first\_loop = 0)  then  first\_loop := 1;  min\_snap\_id := c1\_rec.v\_snap\_id ;  end if ;  select  min(A.snap\_id)  into  snap\_id2  from  dba\_hist\_snapshot A, v$database B  where  A.snap\_id > c1\_rec.v\_snap\_id  and A.dbid=B.dbid;  if (snap\_id2 is null)  then  exit ;  end if ;    select count(\*) into snap\_bidon from dba\_hist\_snapshot where snap\_id=c1\_rec.v\_snap\_id;    if ( snap\_bidon = 0 )  then  null;  else  select  count(\*)  into  startupdb  from  v$instance A, sys.wrm$\_snapshot B, sys.wrm$\_snapshot C  where  B.snap\_id=c1\_rec.v\_snap\_id and  C.snap\_id=snap\_id2 and  A.startup\_time between B.begin\_interval\_time and C.begin\_interval\_time;    if ( startupdb = '0' )  then  filereportzip := 'odp'||'\_'|| '~v\_date' ||'\_'||instance||'.zip';  filereport := 'odp'||'\_'|| '~v\_date' || '\_'||c1\_rec.v\_snap\_id||'\_'|| snap\_id2||'.txt';  dbms\_output.put\_line('define begin\_snap='||c1\_rec.v\_snap\_id);  dbms\_output.put\_line('define end\_snap='||snap\_id2);  dbms\_output.put\_line('define report\_type=text');  dbms\_output.put\_line('define num\_days=7');  if (substr(portstring,1,5) = 'IBMPC') then  dbms\_output.put\_line('define report\_name=tmp\_odp\'||filereport);  dbms\_output.put\_line('@%ORACLE\_HOME%\rdbms\admin\awrrpt');  else  dbms\_output.put\_line('define report\_name=tmp\_odp/'||filereport);  dbms\_output.put\_line('@$ORACLE\_HOME/rdbms/admin/awrrpt');  end if ;  end if;  end if;  end loop;  if ( filereportzip is not null ) then  if (substr(portstring,1,5) = 'IBMPC') then  dbms\_output.put\_line('set define ~');  dbms\_output.put\_line('host cd tmp\_odp & zip '||filereportzip || ' ' || filereport2 );  dbms\_output.put\_line('host cd tmp\_odp & del '|| filereport2 );  else  dbms\_output.put\_line('host cd tmp\_odp ; zip '||filereportzip || ' ' || filereport2 );  dbms\_output.put\_line('host cd tmp\_odp ; rm -f '|| filereport2 );  end if ;  dbms\_output.put\_line('prompt Result is in directory [ tmp\_odp ]:');  dbms\_output.put\_line('prompt ==================================');  if (substr(portstring,1,5) = 'IBMPC') then  dbms\_output.put\_line('host cd tmp\_odp & dir '||filereportzip);  else  dbms\_output.put\_line('host cd tmp\_odp; ls -l '||filereportzip);  end if ;  else  dbms\_output.put\_line('--');  dbms\_output.put\_line('-- ERROR: no enough snap\_id found for date [ ~v\_date ]!');  dbms\_output.put\_line('--');  end if;  exception  when no\_data\_found then  dbms\_output.put\_line('--');  dbms\_output.put\_line('-- ERROR: no enough snap\_id found for date [ ~v\_date ]!');  dbms\_output.put\_line('--');  when others then  dbms\_output.put\_line('--');  dbms\_output.put\_line('-- ERROR: '||sqlerrm);  dbms\_output.put\_line('--');  end;  /  spool off  set define &  @tmp\_odp/tmp\_awrondataperf.lst  exit |
| AWR\_PCREPORT.SQL | define num\_days = 0;  define report\_type = 'html'  column inst\_num new\_value inst\_num  column dbname new\_value dbname  column dbid new\_value dbid  column begin\_snap new\_value begin\_snap  column end\_snap new\_value end\_snap  column report\_name new\_value report\_name  column instance\_name new\_value instance\_name  SELECT &1 dbid  FROM dual;  SELECT &2 inst\_num  FROM dual;  SELECT &3 begin\_snap  FROM dual;  SELECT &4 end\_snap  FROM dual;  SELECT name  ||''  ||'&5' report\_name  FROM v$database;  @@?/rdbms/admin/awrrpti |
| AWR\_GEN\_BATCH\_DAYS.SQL | set echo off heading off feedback off verify off  select 'You have entered num of days:', '&&numdays'from dual;  set linesize 400  set pagesize 10000  spool batch.sql  SELECT DISTINCT '@./rpthtml/awr\_pcreport '||b.dbid  ||' '  ||b.instance\_number  ||' '  ||b.snap\_id  ||' '  ||e.snap\_id  ||' '  ||b.instance\_number  ||'\_AWR\_'  ||TO\_CHAR(b.end\_interval\_time,'YYMMDD\_HH24MI\_')  ||TO\_CHAR(e.end\_interval\_time,'HH24MI')  ||'.html' Commands,  '-- '||TO\_CHAR(b.end\_interval\_time,'YYMMDD\_HH24MI') lineorder  FROM dba\_hist\_snapshot b,  dba\_hist\_snapshot e  WHERE b.end\_interval\_time > trunc(sysdate - &&numdays)  AND e.snap\_id =b.snap\_id+1  ORDER BY lineorder  /  spool off  @batch.sql  column timecol new\_value timestamp  select to\_char(sysdate,'DD\_MON\_YYYY\_HH24') timecol from sys.dual;  column output new\_value dbname  select value || '\_' output from v$parameter where name = 'db\_name';  host zip AWR\_&&dbname&&timestamp \*.html  host rm \*.html  host rm batch.sql |
| HEALTHCHECK\_v1.0 | -- HPT Vietnam Corporation - Data Solution Department  -- Version: 1.0  -- Edited: 31/01/2014    set feed off markup html on spool on  alter session set nls\_date\_format='YYYY-MM-DD';  set pageline 50000  set pagesize 50000  col fn new\_value filename  select 'pfcheck\_'||to\_char(sysdate, 'ddmmyyyy\_hh24.mi')||'.xls' as fn from dual;  spool &filename  prompt ==================================================  prompt Database Name  prompt ==================================================  prompt  select name from v$database;  prompt ==================================================  prompt Database size  prompt ==================================================  prompt  col "Database Size" format a20  col "Free space" format a20  col "Used space" format a20  select round(sum(used.bytes) / 1024 / 1024 / 1024 ) || ' GB' "Database Size"  , round(sum(used.bytes) / 1024 / 1024 / 1024 ) -  round(free.p / 1024 / 1024 / 1024) || ' GB' "Used space"  , round(free.p / 1024 / 1024 / 1024) || ' GB' "Free space"  from (select bytes  from v$datafile  union all  select bytes  from v$tempfile  union all  select bytes  from v$log) used  , (select sum(bytes) as p  from dba\_free\_space) free  group by free.p  /  prompt ==================================================  prompt Tablespace Usage  prompt ==================================================  prompt  SELECT /\* + RULE \*/ df.tablespace\_name "Tablespace",  df.bytes / (1024 \* 1024) "Size (MB)",  SUM(fs.bytes) / (1024 \* 1024) "Free (MB)",  Nvl(Round(SUM(fs.bytes) \* 100 / df.bytes),1) "% Free",  Round((df.bytes - SUM(fs.bytes)) \* 100 / df.bytes) "% Used"  FROM dba\_free\_space fs,  (SELECT tablespace\_name,SUM(bytes) bytes  FROM dba\_data\_files  GROUP BY tablespace\_name) df  WHERE fs.tablespace\_name (+) = df.tablespace\_name  GROUP BY df.tablespace\_name,df.bytes  UNION ALL  SELECT /\* + RULE \*/ df.tablespace\_name tspace,  fs.bytes / (1024 \* 1024),  SUM(df.bytes\_free) / (1024 \* 1024),  Nvl(Round((SUM(fs.bytes) - df.bytes\_used) \* 100 / fs.bytes), 1),  Round((SUM(fs.bytes) - df.bytes\_free) \* 100 / fs.bytes)  FROM dba\_temp\_files fs,  (SELECT tablespace\_name,bytes\_free,bytes\_used  FROM v$temp\_space\_header  GROUP BY tablespace\_name,bytes\_free,bytes\_used) df  WHERE fs.tablespace\_name (+) = df.tablespace\_name  GROUP BY df.tablespace\_name,fs.bytes,df.bytes\_free,df.bytes\_used  ORDER BY 4 DESC;  prompt ==================================================  prompt Top 10 Biggest Table  prompt ==================================================  prompt  column SEGMENT\_NAME format A50  SELECT \* FROM (  SELECT  OWNER, SEGMENT\_NAME, BYTES/1024/1024 SIZE\_MB  FROM  DBA\_SEGMENTS  WHERE  SEGMENT\_TYPE = 'TABLE'  ORDER BY  BYTES/1024/1024 DESC ) WHERE ROWNUM <= 10;  prompt ==================================================  prompt Top 10 Biggest Index  prompt ==================================================  prompt  column SEGMENT\_NAME format A50  SELECT \* FROM (  SELECT  OWNER, SEGMENT\_NAME, BYTES/1024/1024 SIZE\_MB  FROM  DBA\_SEGMENTS  WHERE  SEGMENT\_TYPE = 'INDEX'  ORDER BY  BYTES/1024/1024 DESC ) WHERE ROWNUM <= 10  /  prompt ==================================================  prompt Top 10 Biggest Segment (for Partitioned Object)  prompt ==================================================  prompt  SELECT \* FROM (  SELECT  SEGMENT\_NAME, SEGMENT\_TYPE, SUM(BYTES)/1024/1024 SIZE\_MB  FROM  DBA\_SEGMENTS  WHERE  SEGMENT\_TYPE LIKE '%PARTI%'  GROUP BY  SEGMENT\_NAME, SEGMENT\_TYPE  ORDER BY  SUM(BYTES)/1024/1024 DESC ) WHERE ROWNUM <= 10  /  prompt ==================================================  prompt Separation of Tables and Indexes into Different Tablespaces  prompt ==================================================  prompt  select \* from  (select owner, tablespace\_name,segment\_name,segment\_type from dba\_segments where owner in (select username from app\_users) order by owner, tablespace\_name)  pivot (COUNT(segment\_name) for (segment\_type) in ('TABLE' as Tables,'INDEX' as Indexes));  prompt ===============================================  prompt Users with Objects in Tablespace SYSTEM, SYSAUX  prompt ===============================================  prompt  col OWNER format a30  col segment\_name format a40  col tablespace\_name format a30  col bytes format a20  select owner "User", segment\_name "Segment Name", segment\_type "Type", tablespace\_name "TABLESPACE", bytes/1024/1024 "Size (MB)"  from dba\_segments  where tablespace\_name in ('SYSTEM','SYSAUX') and owner in (select username from app\_users);  prompt ======================  prompt Tables without Indexes  prompt ======================  prompt  SELECT DISTINCT S.OWNER, S.SEGMENT\_NAME,S.BYTES/1024/1024 "Size (MB)" FROM DBA\_SEGMENTS S LEFT JOIN DBA\_INDEXES I  ON (S.SEGMENT\_NAME = I.TABLE\_NAME)  WHERE S.SEGMENT\_TYPE LIKE '%TABLE%'  AND S.OWNER IN (SELECT USERNAME FROM APP\_USERS)  ORDER BY S.OWNER,S.SEGMENT\_NAME;  prompt ==================================  prompt Tables with More Than Five Indexes  prompt ==================================  prompt  select Owner, TABLE\_NAME "Table", COUNT(\*) "Number of Indexes"  from dba\_indexes  where OWNER in (select username from app\_users)  group by OWNER, TABLE\_NAME  having COUNT(\*) > 5  order by COUNT(\*) desc, OWNER, TABLE\_NAME;  prompt ======================================  prompt Tables without Primary Key Constraints  prompt ======================================  prompt  select OWNER,count(TABLE\_NAME) "Number of Tables"  from dba\_tables dt  where not exists (  select 'TRUE'  from dba\_constraints dc  where dc.TABLE\_NAME = dt.TABLE\_NAME  and dc.CONSTRAINT\_TYPE='P')  and OWNER in (select username from app\_users)  group by owner  order by OWNER;  prompt  select OWNER, TABLE\_NAME  from dba\_tables dt  where not exists (  select 'TRUE'  from dba\_constraints dc  where dc.TABLE\_NAME = dt.TABLE\_NAME  and dc.CONSTRAINT\_TYPE='P')  and OWNER in (select username from app\_users)  order by OWNER;  prompt ==============================  prompt Tables or Indexes Not Analyzed  prompt ==============================  prompt  select owner, count(last\_analyzed) "Analyzed", count(\*) - count(last\_analyzed) "Not Analyzed", decode(min(last\_analyzed),NULL,'Never Analyze',to\_char(min(last\_analyzed),'dd-Mon-YYYY')) "Oldest Analyzed", count(\*) "Total Tables"  from dba\_tables  where owner in (select username from app\_users)  group by owner  order by owner;  prompt  select owner, count(last\_analyzed) "Analyzed", count(\*) - count(last\_analyzed) "Not Analyzed", decode(min(last\_analyzed),NULL,'Never Analyze',to\_char(min(last\_analyzed),'dd-Mon-YYYY')) "Oldest Analyzed", count(\*) "Total Tables"  from dba\_indexes  where owner in (select username from app\_users)  group by owner  order by owner;  prompt ==============================  prompt Partitioned Indexes and Tables  prompt ==============================  prompt  SELECT OWNER "Table Owner", TABLE\_NAME "Table", PARTITIONING\_TYPE "Partitioning Type", PARTITION\_COUNT "Partition Count", SUBPARTITIONING\_TYPE "Subpartition Type", DEF\_SUBPARTITION\_COUNT " Subpartition Count"  FROM ALL\_PART\_TABLES  WHERE OWNER IN (SELECT USERNAME FROM APP\_USERS);  prompt  Select OWNER "Owner", INDEX\_NAME "Index", TABLE\_NAME "Table", PARTITIONING\_TYPE "Partioning Type", PARTITION\_COUNT "Partition Count", SUBPARTITIONING\_TYPE "Sub partitioning Type", DEF\_SUBPARTITION\_COUNT "Sub partition Count", LOCALITY "Locality", DEF\_TABLESPACE\_NAME "Default Tablespace"  from DBA\_PART\_INDEXES  WHERE OWNER IN (SELECT USERNAME FROM APP\_USERS);  prompt ==============================  prompt Invalid Objects  prompt ==============================  prompt  Select owner "Owner", object\_name "Name", object\_type "Type", status "Status", created "Created", last\_ddl\_time "Last DDL Time"  from all\_objects  where status!='VALID';  prompt ==============================  prompt Database Links  prompt ==============================  prompt  select owner "Owner", db\_link "Database Link", username "Username", host "Host", created "Created" from dba\_db\_links order by owner, db\_link;  prompt ==============================  prompt Database Jobs  prompt ==============================  prompt  select name,value,isdefault from v$parameter where name = 'job\_queue\_processes';  prompt  select job "Id", log\_user "Submitter", what "Job", last\_date "Last Ok Date", next\_date "Next Run Time", failures "Errors", broken "Broken" from dba\_jobs order by log\_user,job;  prompt ==============================  prompt Large Objects  prompt ==============================  prompt  select owner "Owner", table\_name "Table", column\_name "Column", tablespace\_name "Tablespace", segment\_name "Segment", index\_name "Index", chunk "Chunk (KB)", cache "Cache", in\_row "In Row", retention "Retention", pctversion "PCT Ver", logging "Logging", partitioned "Partitioned" from dba\_lobs where owner in (select username from app\_users) order by owner,table\_name;  prompt ==============================  prompt Objects with NOLOGGING Option  prompt ==============================  prompt  select force\_logging "Force Logging" from v$database;  prompt  select owner "Owner", table\_name "Table", decode(logging,NULL,'NO',logging) "Logging" from dba\_tables where owner in (select username from app\_users) order by owner,table\_name;  prompt  select owner "Owner", index\_name "Index", table\_name "Table", decode(logging,NULL,'NO',logging) "Logging" from dba\_indexes where owner in (select username from app\_users) order by owner,table\_name;  prompt ==============================  prompt Dropped objects  prompt ==============================  prompt  Show parameter recyclebin  prompt  select object\_name "Object Name", original\_name "Ori Name", owner "Owner", type "Type", createtime "Created Time", droptime "Drop Time", space "Number of blocks" from dba\_recyclebin;  prompt  prompt  spool off  set markup html off spool off |
| RDA.sh | #!/bin/sh  ##################################################  # rda.sh: Shell Script Wrapper for RDA  #  # $Id: rda.sh,v 1.15 2014/11/04 09:54:11 RDA Exp $  # ARCS: $Header: /home/cvs/cvs/RDA\_8/src/scripting/bin/rda.sh,v 1.15 2014/11/04 09:54:11 RDA Exp $  #  # Copyright (c) 2002, 2015, Oracle and/or its affiliates. All rights reserved.  #  # Change History  # 20141104 MSC Eliminate letter range.  ##################################################  # Determine the name of the RDA compiled engine  find\_ce\_name()  {  OS=`uname -s | tr 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' \  'abcdefghijklmnopqrstuvwxyz'`  [ -z "$OS" ] && return  case $OS in  aix|darwin) RDA\_ENG="rda\_$OS"  ;;  hp-ux) RDA\_ENG="rda\_hpux"  ;;  linux) case `uname -p` in  386|486|586|686|x86\_64) RDA\_ENG="rda\_linux"  ;;  esac  ;;  osf1) RDA\_ENG="rda\_osf"  ;;  sunos) case `uname -p` in  sparc) RDA\_ENG="rda\_sunos"  ;;  esac  ;;  esac  }  # Detect the RDA software directory  if [ -z "$RDA\_DIR" -o ! -d "$RDA\_DIR" -o ! -f "$RDA\_DIR/rda.pl" ]  then  RDA\_ALTER\_RDA\_DIR="RDA\_DIR=$RDA\_DIR"  export RDA\_ALTER\_RDA\_DIR  unset RDA\_DIR  RDA\_PGM="$0"  if [ -h "$RDA\_PGM" ]  then  RDA\_DIR=`dirname "$RDA\_PGM"`  RDA\_PGM=`ls -l "$RDA\_PGM" 2>/dev/null | sed -e 's/^.\*-> //'`  expr "$RDA\_PGM" : '\/' >/dev/null 2>&1 || RDA\_PGM="$RDA\_DIR/$RDA\_PGM"  fi  RDA\_DIR=`dirname "$RDA\_PGM"`  fi  # Force Perl taint mode  if [ -z "$RDA\_ALTER\_PERL5OPT" ]  then  RDA\_ALTER\_PERL5DIR="PERL5DIR=$PERL5DIR"  RDA\_ALTER\_PERL5INC="PERL5INC=$PERL5INC"  RDA\_ALTER\_PERL5LIB="PERL5LIB=$PERL5LIB"  RDA\_ALTER\_PERL5OPT="PERL5OPT=$PERL5OPT"  RDA\_ALTER\_PERL\_EXE="PERL\_EXE=$PERL\_EXE"  export RDA\_ALTER\_PERL5DIR RDA\_ALTER\_PERL5INC RDA\_ALTER\_PERL5LIB \  RDA\_ALTER\_PERL5OPT RDA\_ALTER\_PERL\_EXE  fi  PERL5OPT='-T'  export PATH PERL5OPT  unset PERL5DIR  unset PERL5INC  unset PERL5LIB  unset PERL\_EXE  # Detect if a compile engine must be used  if [ -r "./rda.cfg" ]  then  . "./rda.cfg"  RDA\_EXE="./${RDA\_EXE:-rda.exe}"  if [ -n "$RDA\_ENG" -a -x "$RDA\_DIR/engine/$RDA\_ENG" ]  then  RDA\_ENG="$RDA\_DIR/engine/$RDA\_ENG"  "$RDA\_ENG" -e"CFG/B\_NO\_CHECK=1" -XUpgrade -- engine "$RDA\_EXE" "$RDA\_ENG"  fi  if [ -x "$RDA\_EXE" ]  then  export RDA\_DIR  exec "$RDA\_EXE" "$@"  fi  elif [ -r "$RDA\_DIR/engine/rda.cfg" ]  then  . "$RDA\_DIR/engine/rda.cfg"  RDA\_EXE="$RDA\_DIR/${RDA\_EXE:-rda.exe}"  if [ -n "$RDA\_ENG" -a -x "$RDA\_DIR/engine/$RDA\_ENG" ]  then  RDA\_ENG="$RDA\_DIR/engine/$RDA\_ENG"  "$RDA\_ENG" -e"CFG/B\_NO\_CHECK=1" -XUpgrade -- engine "$RDA\_EXE" "$RDA\_ENG"  fi  if [ -x "$RDA\_EXE" ]  then  exec "$RDA\_EXE" "$@"  fi  fi  unset PERL5OPT  # Check for a native Perl  if [ -n "$RDA\_NATIVE\_PERL" -a -x "$RDA\_NATIVE\_PERL" ]  then  PERL\_EXE="$RDA\_NATIVE\_PERL"  elif [ -z "$RDA\_NO\_NATIVE" ]  then  PERL\_CMD=`(unset LANG; LC\_ALL=C type perl) 2>/dev/null`  PERL\_EXE=`expr "${PERL\_CMD}" : "perl is a tracked alias for \(.\*\)"`  if [ -z "${PERL\_EXE}" ]  then  PERL\_EXE=`expr "$PERL\_CMD" : "perl is hashed (\(.\*\))"`  if [ -z "$PERL\_EXE" ]  then  PERL\_EXE=`expr "$PERL\_CMD" : "perl is \(.\*\)"`  fi  fi  fi  # Validate the local Perl  if [ -n "$PERL\_EXE" ]  then  ( ulimit -c 0  "$PERL\_EXE" -T -e "die 'too old' if $] < 5.005; use strict"  ) >/dev/null 2>/dev/null  if [ $? -ne 0 ]  then  PERL\_EXE=''  fi  fi  # Validate Applications Perl  if [ -z "$PERL\_EXE" -a -n "$ADPERLPRG" ]  then  ( ulimit -c 0  "$ADPERLPRG" -T -e "die 'too old' if $] < 5.005; use strict"  ) >/dev/null 2>/dev/null  if [ $? -eq 0 ]  then  PERL\_EXE="$ADPERLPRG"  fi  fi  # Validate Perl in RDA .config file  if [ -z "$PERL\_EXE" -a -r .config ]  then  . ./.config  ( ulimit -c 0  "$PERL5DIR"/perl $PERL5INC -T -e "die 'too old' if $] < 5.005; use strict"  ) >/dev/null 2>/dev/null  if [ $? -eq 0 ]  then  PERL\_EXE="$PERL5DIR"/perl  fi  fi  # Locate Perl  ORA\_HOME="${IAS\_ORACLE\_HOME:-$ORACLE\_HOME}"  if [ -z "$PERL\_EXE" ]  then  # Locate Perl in the Oracle home  if [ -n "$ORA\_HOME" ]  then  if [ -x "$ORA\_HOME"/perl/bin/perl ]  then  PERL\_EXE=`echo "$ORA\_HOME"/perl/bin/perl`  PERL\_LIB=`echo "$ORA\_HOME"/perl/lib`  PERL\_SHL=`echo "$ORA\_HOME"/lib`  elif [ -x "$ORA\_HOME"/perl/5\*/bin/perl ]  then  PERL\_EXE=`echo "$ORA\_HOME"/perl/5\*/bin/perl`  PERL\_LIB=`echo "$ORA\_HOME"/perl/5\*/lib`  PERL\_SHL=`echo "$ORA\_HOME"/lib`  elif [ -x "$ORA\_HOME"/Apache/perl/bin/perl ]  then  PERL\_EXE=`echo "$ORA\_HOME"/Apache/perl/bin/perl`  PERL\_LIB=`echo "$ORA\_HOME"/Apache/perl/lib`  PERL\_SHL=`echo "$ORA\_HOME"/lib`  elif [ -x "$ORA\_HOME"/Apache/perl/5\*/bin/perl ]  then  PERL\_EXE=`echo "$ORA\_HOME"/Apache/perl/5\*/bin/perl`  PERL\_LIB=`echo "$ORA\_HOME"/Apache/perl/5\*/lib`  PERL\_SHL=`echo "$ORA\_HOME"/lib`  elif [ -x "$ORA\_HOME"/Apache/perl/5\*/bin/\*/perl ]  then  PERL\_EXE=`echo "$ORA\_HOME"/Apache/perl/5\*/bin/\*/perl`  PERL\_LIB=`echo "$ORA\_HOME"/Apache/perl/5\*/lib`  PERL\_SHL=`echo "$ORA\_HOME"/lib`  fi  fi  # Locate Perl in OCM  if [ -z "$PERL\_EXE" ]  then  # Locate OCM  CCR\_PERL=''  if [ -d ../ccr/engines/\*/perl ]  then  CCR\_PERL=`echo ../ccr/engines/\*/perl`  elif [ -n "$ORA\_HOME" -a -d "$ORA\_HOME"/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$ORA\_HOME"/ccr/engines/\*/perl`  elif [ -n "$ORA\_HOME" -a \  -d "$ORA\_HOME"/../oracle\_common/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$ORA\_HOME"/../oracle\_common/ccr/engines/\*/perl`  elif [ -n "$ORA\_HOME" -a -d "$ORA\_HOME"/../utils/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$ORA\_HOME"/../utils/ccr/engines/\*/perl`  elif [ -n "$MW\_HOME" -a -d "$MW\_HOME"/oracle\_common/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$MW\_HOME"/oracle\_common/ccr/engines/\*/perl`  elif [ -n "$MW\_HOME" -a -d "$MW\_HOME"/utils/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$MW\_HOME"/utils/ccr/engines/\*/perl`  elif [ -n "$WL\_HOME" -a -d "$WL\_HOME"/../oracle\_common/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$WL\_HOME"/../oracle\_common/ccr/engines/\*/perl`  elif [ -n "$WL\_HOME" -a -d "$WL\_HOME"/../utils/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo "$WL\_HOME"/../utils/ccr/engines/\*/perl`  elif [ -d /usr/lib/ccr/engines/\*/perl ]  then  CCR\_PERL=`echo /usr/lib/ccr/engines/\*/perl`  elif [ -n "${ORACLE\_CONFIG\_HOME}" ]  then  CCR\_PROP="$ORACLE\_CONFIG\_HOME"/ccr/config/collector.properties  if [ -r "$CCR\_PROP" ]  then  CCR\_HOME=`grep -e "^ccr.binHome=" "$CCR\_PROP" | cut -d= -f 2`  if [ -d "$CCR\_HOME"/engines/\*/perl ]  then  CCR\_PERL=`echo "$CCR\_HOME"/engines/\*/perl`  fi  fi  fi  # Locate OCM Perl  if [ -n "$CCR\_PERL" ]  then  if [ -x "$CCR\_PERL"/bin/perl ]  then  PERL\_EXE=`echo "$CCR\_PERL"/bin/perl`  PERL\_LIB=`echo "$CCR\_PERL"/lib`  elif [ -x "$CCR\_PERL"/5\*/bin/\*/perl ]  then  PERL\_EXE=`echo "$CCR\_PERL"/5\*/bin/\*/perl`  PERL\_LIB=`echo "$CCR\_PERL"`  fi  fi  fi  # Validate the Perl found  if [ -n "$PERL\_EXE" ]  then  PERL5DIR=`dirname "$PERL\_EXE"`  PERL5INC=''  if [ -d "$PERL\_LIB" ]  then  for DIR in `find "$PERL\_LIB" -type d -name auto -exec dirname '{}' \;`  do  PERL5INC="-I$DIR $PERL5INC"  done  fi  "$PERL\_EXE" $PERL5INC -T -e "die 'too old' if $] < 5.005; use strict" \  >/dev/null 2>/dev/null  if [ $? -ne 0 ]  then  PERL5DIR=''  elif [ -w . ]  then  echo "PERL5DIR='$PERL5DIR'" >.config  echo "PERL5INC='$PERL5INC'" >>.config  if [ -n "$PERL\_SHL" ]  then  echo "PERL\_SHL='$PERL\_SHL'" >>.config  fi  fi  fi  # Try to use compiled engine  if [ -z "$PERL5DIR" ]  then  # Try to configure and use a compiled engine  RDA\_ENG=''  if [ -w '.' -a ! -f 'rda.cfg' -a ! -f 'rda.exe' ]  then  find\_ce\_name  if [ -n "$RDA\_ENG" ]  then  if [ -x "$RDA\_DIR/engine/$RDA\_ENG" -a -r "$RDA\_DIR/engine/$RDA\_ENG" ]  then  cp "$RDA\_DIR/engine/$RDA\_ENG" rda.exe  echo "RDA\_ENG=\"$RDA\_ENG\"" > rda.cfg  echo "RDA\_EXE=\"rda.exe\"" >> rda.cfg  echo "D\_RDA=\"$RDA\_DIR\"" >> rda.cfg  PERL5OPT='-T'  export PERL5OPT  exec ./rda.exe "$@"  fi  fi  fi  # Give appropriate error message  if [ -z "$ORA\_HOME" ]  then  echo "Error: ORACLE\_HOME is not set."  echo "Please set your ORACLE\_HOME."  else  echo "Error: Perl not found in the PATH or in known directory locations."  if [ -n "$RDA\_ENG" ]  then  echo "Although the default RDA engine requires Perl, a compiled version"  echo "without Perl requirements is available for this platform."  echo "Download the platform-specific RDA engine from My Oracle Support"  echo "and place it within the top directory of your RDA installation."  fi  fi  exit 1  fi  fi  # Extend the shared library path  if [ -n "$PERL\_SHL" ]  then  OS=`"$PERL\_EXE" $PERL5INC -e 'print $^O;'`  if [ "$OS" = "aix" ]  then  RDA\_ALTER\_SHL="LIBPATH=$LIBPATH"  if [ -z "$LIBPATH" ]  then  LIBPATH="$PERL\_SHL"  else  LIBPATH="$PERL\_SHL:$LIBPATH"  fi  export LIBPATH RDA\_ALTER\_SHL  elif [ "$OS" = "hpux" ]  then  RDA\_ALTER\_SHL="SHLIB\_PATH=$SHLIB\_PATH"  if [ -z "$SHLIB\_PATH" ]  then  SHLIB\_PATH="$PERL\_SHL"  else  SHLIB\_PATH="$PERL\_SHL:$SHLIB\_PATH"  fi  export SHLIB\_PATH RDA\_ALTER\_SHL  elif [ "$OS" = "darwin" ]  then  RDA\_ALTER\_SHL="DYLD\_LIBRARY\_PATH=$DYLD\_LIBRARY\_PATH"  if [ -z "$DYLD\_LIBRARY\_PATH" ]  then  DYLD\_LIBRARY\_PATH="$PERL\_SHL"  else  DYLD\_LIBRARY\_PATH="$PERL\_SHL:$DYLD\_LIBRARY\_PATH"  fi  export DYLD\_LIBRARY\_PATH RDA\_ALTER\_SHL  else  RDA\_ALTER\_SHL="LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH"  if [ -z "$LD\_LIBRARY\_PATH" ]  then  LD\_LIBRARY\_PATH="$PERL\_SHL"  else  LD\_LIBRARY\_PATH="$PERL\_SHL:$LD\_LIBRARY\_PATH"  fi  export LD\_LIBRARY\_PATH RDA\_ALTER\_SHL  fi  fi  # Test if DBD::Oracle can be used  if [ -z "$RDA\_NO\_DBD\_ORACLE" ]  then  "$PERL\_EXE" $PERL5INC -T -e "use DBI; use DBD::Oracle;" >/dev/null 2>/dev/null  if [ $? -ne 0 ]  then  RDA\_NO\_DBD\_ORACLE=1  export RDA\_NO\_DBD\_ORACLE  fi  fi  # Run the Perl script  exec "${PERL\_EXE:-perl}" $PERL5INC -T "$RDA\_DIR/rda.pl" "$@" |