```
#!/usr/bin/ksh
# Program : rman_online_backup.sh
# Date    : 01-DEC-19
# Author  : Sean Francis
# Purpose : To run an RMAN full database backup plus unbacked up archived
#           redo logs (deleting them after backup) for either a non-CDB, a
#           CDB or a PDB. In addition, all backups are cross checked and
#           obsolete back ups deleted.
#
# Notes   : Command line parameter usage:
#           1 = -NONCDB or -CDB or -LOGS
#           2 = <NON_CDB_NAME> or <CDB_NAME>
#           3 = -PDB (optional)
#           4 = <PDB_NAME> (optional)
#
#           This script will not backup a database which is either down or not
#           in archive log mode.
#
################################################################################
# Who                   When      What
# --------------------- --------- ---------------------------------------------
# S.Francis             01-DEC-19 Added documentation blocks.
################################################################################
# save the arguments so the functions can see them
ARG1=${1}
ARG2=${2}
ARG3=${3}
ARG4=${4}

# global variables and constants
DATE=`date +%F`
DATE_TIME=`date +%F_%H:%M`
LOCAL_BIN=/usr/local/bin
ORATAB=/etc/oratab
BACKUP_HOME=/nas/backups
SCRIPT_HOME=/nas/scripts/backup

# RMAN credentials
RMAN_USER=rmanbackup
RMAN_USER_PWD=rmanbackup
RMAN_CATALOG_USER=rco
RMAN_CATALOG_USER_PWD=rco
RMAN_CATALOG_TNS=rmancat

################################################################################
# function: print_usage
#
# Just prints the script usage to the screen.
################################################################################
function print_usage
{
  print ""
  print "rman_backup.sh: Usage (must be run as the oracle user)"
  print "  rman_online_backup.sh -help                                Displays this help message"
  print "  rman_online_backup.sh -NONCDB <NON_CDB_NAME>               Runs a full non-CDB backup"
  print "  rman_online_backup.sh -CDB <CDB_NAME>                      Runs a full CDB backup plus all PDBs"
  print "  rman_online_backup.sh -CDB <CDB_NAME> -PDB <PDB_NAME>      Runs a full PDB backup"
  print "  rman_online_backup.sh -LOGS <NON_CDB_NAME> | <CDB_NAME>    Runs an archived redo log backup"
  print ""
}

################################################################################
# function: check_oratab
#
# Checks to see if the oratab file exists in the default location and if so,
# check if there's an entry for argument 2 (ORACLE_SID). If there is, it sets
# the oracle environment.
################################################################################
function check_oratab
{
  oracle_sid_length=0
  sid=${ARG2}

  if [ -e ${ORATAB} ]
  then
    for oratab_sid in `grep -v '^#\|^*\|^$' ${ORATAB} | cut -d ':' -f 1`
    do
      if [ ${oratab_sid} = ${sid} ]
      then
```

```
        ORACLE_SID=${sid}
        oracle_sid_length=`print ${#ORACLE_SID}`
      fi
    done

    if [ ${oracle_sid_length} -ne 0 ]
    then
      export ORACLE_SID
      export ORAENV_ASK=NO
      . ${LOCAL_BIN}/oraenv > /dev/null
    else
      print ""
      echo "  *** ERROR: ${ORATAB} contains no entry for ${sid} on server: ${HOSTNAME}"
      print ""
      return 1
    fi
  else
    print ""
    echo "  *** ERROR: ${ORATAB} does not exist on server: ${HOSTNAME}"
    print ""
    return 1
  fi
}

################################################################################
# function: check_instance
#
# Checks to see if the given instance is up and running by searching for its
# pmon process. Extra care is taken to avoid false positives where, for example,
# checking for instance prod1 which is down but prod12 is up.
################################################################################
function check_instance
{
  is_running=NO

  for ps_str in `ps -ef | grep -v "grep" | grep pmon_${ORACLE_SID} | sed 's/ //g'`
  do
    char_pos=`echo ${ps_str} | awk '{print index($ps_str,"pmon_")}'`
    running_inst=`echo ${ps_str} | cut -c${char_pos}- | cut -d"_" -f2`

    if [ "$running_inst" = "$ORACLE_SID" ]
    then
      is_running=YES
    fi
  done

  if [ ${is_running} = "YES" ]
  then
    return 0
  else
    print ""
    echo "  *** ERROR: ${ORACLE_SID} is not running on server: ${HOSTNAME}"
    print ""
    return 1
  fi
}

################################################################################
# function: get_log_mode
#
# Logs into an instance and queries the log mode of the database.
################################################################################
function get_log_mode
{
  ${ORACLE_HOME}/bin/sqlplus -S /NOLOG << EOF
  conn / as sysdba
  set heading off
  set feedback off
  set verify off
  SELECT LOG_MODE from V\$DATABASE;
  EXIT
EOF
}

################################################################################
# function: check_log_mode
#
# Determines the archivelog mode of the database. Calls get_log_mode.
################################################################################
```

```
function check_log_mode
{
  log_mode=`get_log_mode`

  # strip out white space
  log_mode=`echo ${log_mode}`

  if [ "$log_mode" = "ARCHIVELOG" ]
  then
    return 0
  else
    print ""
    echo "  *** ERROR: ${ORACLE_SID} not in ARCHIVELOG mode - cannot perform online backup"
    print ""
    return 1
 fi
}

################################################################################
# function: check_pdb_exists
#
# Queries the CDB for the existence of the named PDB.
################################################################################
function check_pdb_exists
{
  ${ORACLE_HOME}/bin/sqlplus -S /NOLOG << EOF
  conn / as sysdba
  set heading off
  set feedback off
  set verify off
  SELECT DECODE(COUNT(*),1,'YES','NO') FROM V\$PDBS WHERE NAME = '${PDB_NAME}';
  EXIT
EOF
}

################################################################################
# function: check_pdb
#
# Checks for the existence of the named PDB.
################################################################################
function check_pdb
{
  PDB_NAME=${ARG4}

  pdb_exists=`check_pdb_exists`

  # strip the white space
  pdb_exists=`echo ${pdb_exists}`

  if [ "$pdb_exists" = "YES" ]
  then
    return 0
  else
    print ""
    echo "  *** ERROR: PDB ${PDB_NAME} does not exist"
    print ""
    return 1
 fi
}

################################################################################
# function: backup_cdb_or_noncdb
#
# Runs an RMAN full database backup for a non-CDB or CDB.
################################################################################
function backup_cdb_or_noncdb
{
  BACKUP_DIR=${BACKUP_HOME}/${ORACLE_SID}/${DATE}/${DATE_TIME}
  mkdir -p ${BACKUP_DIR}

  LOG_DIR=${SCRIPT_HOME}/LOGS/${DATE}
  mkdir -p ${LOG_DIR}
  LOG_FILE=${LOG_DIR}/${ORACLE_SID}_${DATE_TIME}_DB_rman.log

  ${ORACLE_HOME}/bin/rman target ${RMAN_USER}/${RMAN_USER_PWD} using sysbackup log=${LOG_FILE} <<EOF
  connect catalog ${RMAN_CATALOG_USER}/${RMAN_CATALOG_USER_PWD}@${RMAN_CATALOG_TNS}
  run
  {
  allocate channel DISK_1 type disk format '${BACKUP_DIR}/%d_%U.bkp';
```

```
  allocate channel DISK_2 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  allocate channel DISK_3 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  backup as compressed backupset database plus archivelog delete input;
  crosscheck backup;
  delete noprompt obsolete;
  release channel DISK_1;
  release channel DISK_2;
  release channel DISK_3;
  }
  exit;
EOF
}

################################################################################
# function: backup_pdb
#
# Runs an RMAN full database backup for a PDB.
################################################################################
function backup_pdb
{
  BACKUP_DIR=${BACKUP_HOME}/${ORACLE_SID}/${PDB_NAME}/${DATE}/${DATE_TIME}
  mkdir -p ${BACKUP_DIR}

  LOG_DIR=${SCRIPT_HOME}/LOGS/${DATE}
  mkdir -p ${LOG_DIR}

  LOG_FILE=${LOG_DIR}/${ORACLE_SID}_${PDB_NAME}_${DATE_TIME}_PDB_rman.log

  ${ORACLE_HOME}/bin/rman target ${RMAN_USER}/${RMAN_USER_PWD} using sysbackup log=${LOG_FILE} <<EOF
  connect catalog ${RMAN_CATALOG_USER}/${RMAN_CATALOG_USER_PWD}@${RMAN_CATALOG_TNS}
  run
  {
  allocate channel DISK_1 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  allocate channel DISK_2 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  allocate channel DISK_3 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  backup pluggable database ${PDB_NAME};
  backup archivelog all not backed up 1 times delete input;
  crosscheck backup;
  delete noprompt obsolete;
  release channel DISK_1;
  release channel DISK_2;
  release channel DISK_3;
  }
  exit;
EOF
}

################################################################################
# function: backup_logs
#
# Runs an RMAN archived redo log backup, deleting them after backup.
################################################################################
function backup_logs
{
  BACKUP_DIR=${BACKUP_HOME}/${ORACLE_SID}/${DATE}/${DATE_TIME}
  mkdir -p ${BACKUP_DIR}

  LOG_DIR=${SCRIPT_HOME}/LOGS/${DATE}
  mkdir -p ${LOG_DIR}
  LOG_FILE=${LOG_DIR}/${ORACLE_SID}_${DATE_TIME}_LOGS_rman.log

  ${ORACLE_HOME}/bin/rman target ${RMAN_USER}/${RMAN_USER_PWD} using sysbackup log=${LOG_FILE} <<EOF
  connect catalog ${RMAN_CATALOG_USER}/${RMAN_CATALOG_USER_PWD}@${RMAN_CATALOG_TNS}
  run
  {
  allocate channel DISK_1 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  allocate channel DISK_2 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  allocate channel DISK_3 type disk format '${BACKUP_DIR}/%d_%U.bkp';
  backup archivelog all not backed up 1 times delete input;
  release channel DISK_1;
  release channel DISK_2;
  release channel DISK_3;
  }
  exit;
EOF
}

################################################################################
# main program
```

```
################################################################################

if [ ${USER} != "oracle" ]
then
  print_usage
  exit 1
fi

#if [ ${#} -eq 0 ] -o [ ${#} -eq 1 ] && [ ${1} = "-help" ]
#then
#  print_usage
#  exit 0
#fi

case ${#} in
  0|1|3 ) print_usage
          exit 0
          ;;
  2     ) case ${ARG1} in
            "-NONCDB"|"-CDB" ) check_oratab
                               if [ ${?} -eq 0 ]
                               then
                                 check_instance
                               else
                                 exit 1
                               fi
                               if [ ${?} -eq 0 ]
                               then
                                 check_log_mode
                               else
                                 exit 1
                               fi
                               if [ ${?} -eq 0 ]
                               then
                                 backup_cdb_or_noncdb
                               else
                                 exit 1
                               fi
                               ;;
            "-LOGS"          ) check_oratab
                               if [ ${?} -eq 0 ]
                               then
                                 check_instance
                               else
                                 exit 1
                               fi
                               if [ ${?} -eq 0 ]
                               then
                                 check_log_mode
                               else
                                 exit 1
                               fi
                               if [ ${?} -eq 0 ]
                               then
                                 backup_logs
                               else
                                 exit 1
                               fi
                               ;;
            *                ) print_usage
                               exit 1
                               ;;
          esac
          ;;
  4     ) if [ "$ARG1" = "-CDB" ] && [ "$ARG3" = "-PDB" ]
          then
            check_oratab
            if [ ${?} -eq 0 ]
            then
              check_instance
            else
              exit 1
            fi
            if [ ${?} -eq 0 ]
            then
              check_log_mode
            else
              exit 1
            fi
```

```
            if [ ${?} -eq 0 ]
            then
              check_pdb
            else
              exit 1
            fi
            if [ ${?} -eq 0 ]
            then
              backup_pdb
            else
              exit 1
            fi
          else
            print_usage
            exit 1
          fi
          ;;
  *     ) print_usage
          exit 1
          ;;
esac

# eof rman_online_backup.sh
```