Using SYS_CONTEXT function for Auditing Purposes in Oracle

Problem: You might need to develop some simple auditing process that for example whenever somebody updates a record, you capture some information about WHO and FROM WHERE it is done.

Answer: By looking carefully at Oracle documents (www.oracle.com), you will find the SYS_CONTEXT function as your quick solution.

Here is general syntax of this function:

```
SELECT sys_context('USERENV', '<parameter>');
FROM DUAL;
```

At first glance you might say this function is the perfect solution for auditing purposes because it provides session related information like user name or host name but I would remind you that most of return values from this function are meaningless unless the application program sets some specific values for them by using special packages which I will cover it later and as you know in the case of security you cannot expect the intruder or culprit to set those values. Still this function is better than nothing for handy quick database level auditing.

Here are list of the most popular parameters to this function:

Name	Description	Example
ACTION	Returns current action set by application using dbms_application_info package	Select sys_context('USERENV', 'ACTION') from dual;
CLIENT_IDENTIFIER	Returns currently assigned client ID by the application program using dbms_session package	Select sys_context('USERENV','CLIENT_IDENTIFIER') from dual;
CURRENT_SCHEMA	Returns current schema	Select sys_context('USERENV','CURRENT_SCHEMA') from dual;
CURRENT_SQL	Returns first 4KB block of SQL statement	Select sys_context('USERENV','CURRENT_SQL') from dual;
HOST	Returns name of the computer which is connected to the DB	Select sys_context('USERENV','HOSt') from dual;
MODULE	Returns name of the current program which should be set by dbms_application_package	Select sys_context('USERENV','MODULE') from dual;
OS_USER	Returns name of current OS user connected to this session	Select sys_context('USERENV','OS_USER') from dual;
SESSIONID	Returns session ID	Select sys_context('USERENV','SESSIONID') from dual;

List of all valid parameters:

ACTION, AUDITED_CURSORID, AUTHENTICATED_IDENTITY, AUTHENTICATION_DATA, AUTHENTICATION_METHOD, BG_JOB_ID, CLIENT_IDENTIFIER, CLIENT_INFO, CURRENT_BIND, CURRENT_EDITION_ID, CURRENT_EDITION_NAME, CURRENT_SCHEMA, CURRENT_SCHEMAID, CURRENT_SQL, CURRENT_SQL*n,* CURRENT_SQL_LENGTH, DB_DOMAIN, DB_NAME, DB_UNIQUE NAME, ENTRYID, ENTERPRISE_IDENTITY, FG_JOB_ID, GLOBAL_CONTEXT_MEMORY, GLOBAL_UID, HOST, IDENTIFICATION_TYPE, INSTANCE, INSTANCE_NAME, IP_ADDRESS, ISDBA, LANG, LANGUAGE, MODULE, NETWORK_PROTOCOL, NLS_CALENDAR, NLS_CURRENCY, NLS_DATE_FORMAT, NLS_DATE_LANGUAGE, NLS_SORT, NLS_TERRITORY, OS_USER, POLICY_INVOKER, PROXY_ENTERPRISE_IDENTITY, PROXY_GLOBAL_UID, PROXY_USER, PROXY_USERID, SERVER_HOST, SERVICE_NAME, SESSION_USER, SESSION_USERID, SESSIONID, SID, STATEMENTID, TERMINAL,

Special Packages – Functions:
DBMS_APPLICATION_INFO.SET_ACTION
DBMS_SESSION.SET_IDENTIFIER

Here is a sample of auditing trigger for auditing purposes. This trigger logs successful and unsuccessful logon attempts. This trigger dumps data to V$LIGHT_AUDIT_LOGON which is defined at the end.

```
create or replace trigger tal_logon_success after logon on database
 begin
   begin -- insert block
     insert into v$light_audit_logon (
       audit_date,
       db_user_name,
       os_user_name,
       computer_name,
       ip_address,
       module,
       comments
     )
     values (
       sysdate,
        user,
        sys_context('userenv','os_user'),
        sys_context('userenv','host'),
        sys_context('userenv','ip_address'),
        sys_context('userenv','module'),
        'Successful Logon'
     );
     exception when others then
       null;
   end; -- insert block
 end;
/

create or replace trigger tal_logon_failure
after servererror
on database
begin
  if (is_servererror(1017)) then
    begin
      insert into v$light_audit_logon (
```

```
        audit_date,
        db_user_name,
        os_user_name,
        computer_name,
        ip_address,
        module,
        comments
      )
      values (
        sysdate,
          sys_context('userenv','session_user'),
          sys_context('userenv','os_user'),
          sys_context('userenv','host'),
          sys_context('userenv','ip_address'),
          sys_context('userenv','module'),
          'UnSuccessful Logon'
      );
    end; -- insert block
  end if;
end logon_failures;
/

create table v$light_audit_logon (
   audit_date    date,
   db_user_name  varchar2(30),
   os_user_name  varchar2(30),
   computer_name varchar2(50),
   ip_address    varchar2(20),
   module        varchar2(30),
   comments      varchar2(500)
 );
```

You can also develop a stored procedure and call it in INSERT/ UPDATE/ DELETE triggers to log who modified data.

Provided by Ali Khademi (www.khademi.com)
Do not hesitate to share your comments with Ali Khademi : ali at khademi dot com