

## 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](http://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\_SQLn, 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](http://www.khademi.com))

Do not hesitate to share your comments with Ali Khademi : ali at khademi dot com