**You Asked**

<https://technology.amis.nl/2007/10/08/implementing-instrumentation-in-adf-applications-for-enhanced-administration/>

Hi  
  
I have a number of pl/sql package.procedures that are making use of global context variables. I am setting the CLIENT\_ID with DBMS\_SESSION.set\_identifier(my\_client\_id); and can see it has been set with sys\_context('USERENV','CLIENT\_IDENTIFIER') . After I have done the processing I need (determining if a user has logged in and successfully been authorized) I redirect to the application. The application wants to then look at the global contexts to determine if they have been set or not, but for some reason it appears my redirect 'looses' the client\_id. I am doing a redirect like this  
HTP.p('<script type="text/javascript">window.location.href="'||my\_url||'"</script>'); (and have tried a number of variations).   
Could I be losing my client id with this redirect or am I off in the weeds and should be looking somewhere else for the cause of the problem? I think my problem may be that I do not understand what is considered the "current session".  
Thank you for any clues

**and we said...**

A lot depends on if/how connection pooling or similar is setup, because each request may be going to a different session that exists on the database, or even a brand new session (eg if all existing sessions that serve requests were busy).   
  
Hence typically for usage of global contexts, the very first part of \*every\* request is a call to DBMS\_SESSION.set\_identifier. Effectively, its a "Hey, its 'me' I'm back again" announcement back to the database.   
  
If somewhere in your midst (redirect or otherwise) you potentially have a call that does not set the client\_id, then it could be lost, and hence you wont see the global context information you might have been expecting.   
  
Hope that makes sense.   
  
eg

SQL> create or replace context DEMO using p accessed globally;

Context created.

SQL>

SQL> create or replace procedure p(p\_attr varchar2) is

2 begin

3 dbms\_session.set\_context('DEMO','ATTR',p\_attr,client\_id=>sys\_context('USERENV','CLIENT\_IDENTIFIER'));

4 end;

5 /

Procedure created.

SQL> exec dbms\_session.set\_identifier('ME');

PL/SQL procedure successfully completed.

SQL> exec p('MY\_VALUE');

PL/SQL procedure successfully completed.

SQL> set serverout on

SQL> create or replace

2 procedure good\_proc is

3 begin

4 dbms\_session.set\_identifier('ME');

5 dbms\_output.put\_line(sys\_context('DEMO','ATTR'));

6 end;

7 /

Procedure created.

SQL>

SQL> create or replace

2 procedure bad\_proc is

3 begin

4 dbms\_output.put\_line(sys\_context('DEMO','ATTR'));

5 end;

6 /

Procedure created.

So good\_proc always sets the client\_id and bad\_proc does not. Under SOME situations, both will appear to work fine.

SQL> exec good\_proc

MY\_VALUE

PL/SQL procedure successfully completed.

SQL> exec bad\_proc

MY\_VALUE

PL/SQL procedure successfully completed.

SQL>

But bad\_proc only worked because good\_proc ran first, and they were in the same session. If I reconnect ...

SQL> conn scott/tiger

Connected.

SQL> set serverout on

SQL> exec bad\_proc

PL/SQL procedure successfully completed.

SQL> exec good\_proc

MY\_VALUE

PL/SQL procedure successfully completed.

then bad\_proc has lost its way.

**and you rated our response**