E-Trustex Platform

Log backup and cleaning

# Introduction

The goal of this document is to describe how logs are backed up to a specific structure and how the system cleans up old logs.

# Concepts

E-Trustex has 2 main DB schemas:

* eTrustex: to store configurations and electronic document
* eTrustBin: to store operational logs and decisional logs

On eTrustex schema has two views allowing to retrieve Oracle audit logs. These audit logs are cleaned up automatically after one month so it is important to keep a backup. One job moves audit logs to dedicated structures on eTrustBin schema.

Moreover the application logs available in the table ETR\_TB\_LOG are also moved to a dedicated structure in eTrustbin schema and every log older than 90 days are removed automatically (via a job) from eTrustex schema.

# Audit Logs

In eTrustex schema, two view are availables:

* ETR\_VW\_AUDIT\_OBJECT
* ETR\_VW\_AUDIT\_SESSION

One job copies from these views to dedicated structures in eTrustbin.

The two tables in eTrustbin are:

* STAGING\_USER\_AUDIT\_OBJECTS
* STAGING\_USER\_AUDIT\_SESSION

The job (executed in eTrustex schema) is:

* Audit move

On production it is executed every days at 3.30 A.M

That job uses two PL/SQL procedures defined in eTrustbin schema:

* SAVE\_AUDIT\_OBJECTS
* SAVE\_AUDIT\_SESSIONS

# Application Logs

The application logs are available in ETR\_TB\_LOG table in eTrustex schema.

The logs are copied to a dedicated structure ETR\_TB\_LOG in eTrustbin schema by a job named "Move logs to binary schema".

That job uses the PL/SQL procedure SAVE\_LOGS.

It is executed on PROD every day at 3.00 A.M.

Moreover another job cleans up old messages from the log table in eTrustex schema.

That job is called Clean old logs. It uses the PL/SQL procudure CLEAN\_OLD\_LOGS that is present in eTrustex schema.

On PROD it is executed every day at 4.00 A.M.

# Annexes

create or replace PROCEDURE SAVE\_AUDIT\_OBJECTS AS

batch\_size number := 20;

current\_count number := 0;

last\_timestamp timestamp;

cursor audit\_log\_cursor(ts timestamp) is SELECT \* FROM etrustex.ETR\_VW\_AUDIT\_OBJECT WHERE NOT (obj\_name = 'ETR\_TB\_LOG' AND action\_name = 'INSERT') AND timestamp > ts;

begin

dbms\_output.put\_line('script start');

SELECT COALESCE( MAX (timestamp) , SYSDATE - 1000 ) max\_date into last\_timestamp FROM STAGING\_USER\_AUDIT\_OBJECTS;

for log in audit\_log\_cursor(last\_timestamp) loop

current\_count := current\_count +1;

insert into STAGING\_USER\_AUDIT\_OBJECTS(

OS\_USERNAME,USERNAME,USERHOST,TERMINAL,TIMESTAMP,OWNER,OBJ\_NAME,ACTION\_NAME,NEW\_OWNER,NEW\_NAME,SES\_ACTIONS,COMMENT\_TEXT,SESSIONID,ENTRYID,STATEMENTID,RETURNCODE,PRIV\_USED,CLIENT\_ID,

ECONTEXT\_ID,SESSION\_CPU,EXTENDED\_TIMESTAMP,PROXY\_SESSIONID,GLOBAL\_UID,INSTANCE\_NUMBER,OS\_PROCESS,TRANSACTIONID,SCN,SQL\_BIND,SQL\_TEXT,OBJ\_EDITION\_NAME

) values (

log.OS\_USERNAME,log.USERNAME,log.USERHOST,log.TERMINAL,log.TIMESTAMP,log.OWNER,log.OBJ\_NAME,log.ACTION\_NAME,log.NEW\_OWNER,log.NEW\_NAME,log.SES\_ACTIONS,log.COMMENT\_TEXT,log.SESSIONID,

log.ENTRYID,log.STATEMENTID,log.RETURNCODE,log.PRIV\_USED,log.CLIENT\_ID,log.ECONTEXT\_ID,log.SESSION\_CPU,null,log.PROXY\_SESSIONID,log.GLOBAL\_UID,log.INSTANCE\_NUMBER,log.OS\_PROCESS,log.TRANSACTIONID,

log.SCN,log.SQL\_BIND,log.SQL\_TEXT,log.OBJ\_EDITION\_NAME );

if (current\_count = batch\_size) then

current\_count := 0;

commit;

dbms\_output.put\_line('commit batch');

end if;

end loop;

commit;

dbms\_output.put\_line('script stop');

EXCEPTION

WHEN OTHERS then

dbms\_output.put\_line('an error has occured');

rollback;

END SAVE\_AUDIT\_OBJECTS;

create or replace PROCEDURE SAVE\_AUDIT\_SESSIONS AS

batch\_size number := 20;

current\_count number := 0;

last\_timestamp timestamp;

cursor audit\_log\_cursor(ts timestamp) is SELECT \* FROM etrustex.ETR\_VW\_AUDIT\_SESSION WHERE timestamp > ts;

begin

dbms\_output.put\_line('script start');

SELECT COALESCE( MAX (timestamp) , SYSDATE - 1000 ) max\_date into last\_timestamp FROM STAGING\_USER\_AUDIT\_SESSION;

for log in audit\_log\_cursor(last\_timestamp) loop

current\_count := current\_count +1;

insert into STAGING\_USER\_AUDIT\_SESSION (

OS\_USERNAME,USERNAME,USERHOST,TERMINAL,TIMESTAMP,ACTION\_NAME,LOGOFF\_TIME,LOGOFF\_LREAD,LOGOFF\_PREAD,LOGOFF\_LWRITE,

LOGOFF\_DLOCK,SESSIONID,RETURNCODE,CLIENT\_ID,SESSION\_CPU,EXTENDED\_TIMESTAMP,PROXY\_SESSIONID,GLOBAL\_UID,INSTANCE\_NUMBER,OS\_PROCESS

) values (

log.OS\_USERNAME,log.USERNAME,log.USERHOST,log.TERMINAL,log.TIMESTAMP,log.ACTION\_NAME,log.LOGOFF\_TIME,log.LOGOFF\_LREAD,

log.LOGOFF\_PREAD,log.LOGOFF\_LWRITE,log.LOGOFF\_DLOCK,log.SESSIONID,log.RETURNCODE,log.CLIENT\_ID,log.SESSION\_CPU,

null,log.PROXY\_SESSIONID,log.GLOBAL\_UID,log.INSTANCE\_NUMBER,log.OS\_PROCESS );

if (current\_count = batch\_size) then

current\_count := 0;

commit;

dbms\_output.put\_line('commit batch');

end if;

end loop;

commit;

dbms\_output.put\_line('script stop');

EXCEPTION

WHEN OTHERS then

dbms\_output.put\_line('an error has occured');

rollback;

END SAVE\_AUDIT\_SESSIONS;

create or replace PROCEDURE SAVE\_LOGS AS

batch\_size number := 200;

current\_count number := 0;

last\_timestamp timestamp;

cursor log\_cursor(ts timestamp) is Select \* FROM ETRUSTEX.etr\_tb\_log WHERE CRE\_DT > ts AND CRE\_DT < (SYSDATE - 1);

begin

dbms\_output.put\_line('script start');

SELECT COALESCE( MAX(CRE\_DT) , to\_date('19700101','yyyymmdd') ) max\_last\_update into last\_timestamp FROM etr\_tb\_log;

for log in log\_cursor(last\_timestamp) loop

current\_count := current\_count +1;

INSERT INTO ETR\_TB\_LOG ( LOG\_ID, LOG\_TYPE, LOG\_OPERATION, LOG\_DESCRIPTION, LOG\_VALUE, LOG\_LARGE\_VALUE, LOG\_URL\_CONTEXT, LOG\_CORRELATION\_ID, LOG\_DOCUMENT\_ID,

LOG\_DOCUMENT\_TYPE\_CD, LOG\_BUS\_CORRELATION\_ID, LOG\_TRA\_ID, LOG\_MSG\_ID, LOG\_MSG\_BIN\_ID, LOG\_ISSUER\_PTY\_ID, LOG\_SENDER\_PTY\_ID, LOG\_RECEIVER\_PTY\_ID, LOG\_BUSINESS\_DOMAIN,

CRE\_DT, CRE\_ID, MOD\_DT, MOD\_ID, LOG\_MSG\_SIZE, LOG\_AUTH\_IP\_ADDRESS, LOG\_ENITITY\_ID, LOG\_ENTITY, LOG\_MODULE, LOG\_USRNAME, LOG\_USR\_ROLE, LOG\_CLASS )

VALUES (

log.LOG\_ID, log.LOG\_TYPE, log.LOG\_OPERATION, log.LOG\_DESCRIPTION, log.LOG\_VALUE, log.LOG\_LARGE\_VALUE, log.LOG\_URL\_CONTEXT, log.LOG\_CORRELATION\_ID, log.LOG\_DOCUMENT\_ID,

log.LOG\_DOCUMENT\_TYPE\_CD, log.LOG\_BUS\_CORRELATION\_ID, log.LOG\_TRA\_ID, log.LOG\_MSG\_ID, log.LOG\_MSG\_BIN\_ID, log.LOG\_ISSUER\_PTY\_ID, log.LOG\_SENDER\_PTY\_ID, log.LOG\_RECEIVER\_PTY\_ID, log.LOG\_BUSINESS\_DOMAIN,

log.CRE\_DT, log.CRE\_ID, log.MOD\_DT, log.MOD\_ID, log.LOG\_MSG\_SIZE, log.LOG\_AUTH\_IP\_ADDRESS, log.LOG\_ENITITY\_ID, log.LOG\_ENTITY, log.LOG\_MODULE, log.LOG\_USRNAME, log.LOG\_USR\_ROLE, log.LOG\_CLASS

);

if (current\_count = batch\_size) then

current\_count := 0;

commit;

dbms\_output.put\_line('commit batch');

end if;

end loop;

commit;

dbms\_output.put\_line('script stop');

EXCEPTION

WHEN OTHERS then

dbms\_output.put\_line('an error has occured');

rollback;

END SAVE\_LOGS;

create or replace PROCEDURE CLEAN\_OLD\_LOGS AS

-- > 3 months logs deletion script

delete\_count number :=0;

delete\_count\_max number := 100; --batch size

cursor log\_cursor is select log\_id from etr\_tb\_log where cre\_dt<sysdate-90 ;

startTS timestamp := sysdate;

totalCount number :=0;

begin

dbms\_output.put\_line('script start ' || startTS);

for log\_item in log\_cursor loop

delete from etr\_tb\_log where log\_id = log\_item.log\_id;

totalCount := totalCount+1;

delete\_count := delete\_count+1;

if delete\_count = delete\_count\_max then

commit;

delete\_count:=0;

dbms\_output.put\_line('commit block');

end if;

exit when startTS + 1/(48) < sysdate; --30 minutes script

end loop;

commit;

dbms\_output.put\_line(totalCount ||' messages removed');

dbms\_output.put\_line('script stop ' ||sysdate);

EXCEPTION

WHEN OTHERS then

dbms\_output.put\_line('an error has occured');

rollback;

END CLEAN\_OLD\_LOGS;