21 DBMS_APP_CONT

The `DBMS_APP_CONT` package provides an interface to determine if the in-flight transaction on a now unavailable session committed or not, and if the last call on that session completed or not.

> **See Also:**
>
> *Oracle Database Development Guide* for explanations of application continuity and Transaction Guard, and the relationship between these two features:
>
> - "Using Transaction Guard"

This chapter contains the following topics:

- Overview
- Security Model
- Summary of DBMS_APP_CONT Subprograms

DBMS_APP_CONT Overview

The DBMS_APP_CONT package can be used to solve this example issue.

Problem Description

One of the fundamental problems for recovering applications after an outage is that the commit message that is sent back to the client is not durable. If there is a break between the client and the server, the client sees an error message indicating that the communication failed. This error does not inform the application whether the submission executed any commit operations or if a procedural call, ran to completion executing all expected commits and session state changes or failed part way through or yet worse, is still running disconnected from the client.

GET_LTXID_OUTCOME

The purpose of the GET_LTXID_OUTCOME Procedure is to determine if the in-flight transaction on a now unavailable session completed or not. It is used when the original session returned an error due to unavailability. Situations that can cause such session unavailability may occur at the session, instance, server, or network, and result from planned or unplanned outages. When such an outage occurs, the application receives a disconnection error. Such an error provides no insight as to whether the transaction committed. It also does not reveal what the application might have been expecting from that commit if it had returned.

> **See Also:**
>
> *Oracle Database Concepts* for explanation of Logical Transaction ID

DBMS_APP_CONT Security Model

Applications must have the EXECUTE privilege on the DBMS_APP_CONT package.

To grant this privilege, ask your database administrator to run the following SQL statement:

```
GRANT execute on DBMS_APP_CONT to application user ;
```

Summary of DBMS_APP_CONT Subprograms

This topic lists the DBMS_APP_CONT subprograms in alphabetical order and briefly describes them.

Table 21-1 DBMS_APP_CONT Package Subprograms

Subprogram	Description
APPLY_REPLAY_RULE Procedure	This procedure activates a new replay rule that is inherited by any sub-routine unless it is overwritten by a new rule applied to that sub-routine.
GET_LTXID_OUTCOME Procedure	Lets customer applications and third party application servers determine the transactional status of the last session when that session becomes unavailable.
GET_REPLAY_RULES Function	This function returns a list of the replay rules. From the results, you can test if any rule target is replayable with bit and function.
RESET_REPLAY_RULES Procedure	This procedure resets all existing replay rules. When you call this procedure, it clears all locally defined rules in the specified scope and restores them to the original state when they were inherited from their parent scope, or the service if parent scope is not specified.

APPLY_REPLAY_RULE Procedure

This procedure activates a new replay rule that is inherited by any sub-routine unless it is overwritten by a new rule applied to that sub-routine.

A replay rule activated in a PL/SQL block loses its effect when reset_replay_rules is called or the block returns.

> **Note:**
>
> Calling this procedure in a sub-block of a procedure, function, or anonymous block applies the rule to its parent scope. When the sub-block returns, the rule will not lose its effect. Thus, you are not recommended to call this procedure in a sub-block.

Syntax

```
DBMS_APP_CONT.APPLY_REPLAY_RULE (
 replayable IN   BOOLEAN,
 targets    IN   BINARY_INTEGER,
 scope      IN   BINARY_INTEGER DEFAULT DBMS_APP_CONT.SCOPE_CURRENT);
```

Parameters

Table 21-2 *APPLY_REPLAY_RULE Procedure Parameters*

Parameter	Description
replayable	TRUE or FALSE, depending on weather the rule is replayable or not.
targets	Target effects on which this rule is applied. If multiple effects are desired, pass them as bit or as individual effects, such as: dbms_app_cont.side_effects + dbms_app_cont.autonomous_transactions
scope	Scope of the rule, either dbms_app_cont.scope_current (default) or dbms_app_cont.scope_parent.

GET_LTXID_OUTCOME Procedure

This procedure lets customer applications and third party application servers determine the transactional status of the last session when that session becomes unavailable.

Syntax

```
DBMS_APP_CONT.GET_LTXID_OUTCOME (
 client_ltxid          IN    RAW,
 committed             OUT   BOOLEAN,
 user_call_completed   OUT   BOOLEAN)
```

Parameters

Table 21-3 *GET_LTXID_OUTCOME Procedure Parameters*

Parameter	Description
client_ltxid	Client-side logical transaction ID. Obtain the LTXID from the previous failed session using the client driver provided APIs - getLTXID for JDBC, and LogicalTransactionId for ODP.net., and OCI_ATTR_GET with LTXID for OCI.
committed	Returns TRUE if the transaction with the named logical LTXID has COMMITTED. Returns FALSE if the logical LTXID has not COMMITTED. When returning FALSE, the procedure blocks the LTXID from further use so that there is no possibility of previous in-flight work committing this LTXID.
user_call_completed	Whether all information has been returned to the client. Examples of such messages are the number of rows processed when using autocommit or commit on success, parameter and function results when calling PL/SQL, or PL/SQL with more work to do after the COMMIT. Applications that expect to use data returned from the commit in order to function correctly must look at this second parameter.

Exceptions

Table 21-4 GET_LTXID_OUTCOME Procedure Exceptions

Exception	Description
ORA-14950 - SERVER_AHEAD	The server is ahead so the transaction is both an old transaction and one which has already committed. This is an error as the application is passing an older LTXID that is the not the last used for that session. The purpose of GET_LTXID_OUTCOME is to return the current transaction outcome for that session after a recoverable outage.
ORA-14951 - CLIENT_AHEAD	The client is ahead of the server. This can happen if the server has been flashed backed, recovered using media recovery, or is a standby that has opened earlier with data loss.
ORA-14906 - SAME_SESSION	Executing GET_LTXID_OUTCOME is not supported on the session owning the LTXID as it blocks further processing on that session after a recoverable outage.
ORA-14909 - COMMIT_BLOCKED	Your session has been blocked from committing by another user with the same username using GET_LTXID_OUTCOME. GET_LTXID_OUTCOME should only be called on dead sessions. Please check with your application administrator.
ORA-14952 - ERROR	The outcome cannot be determined. During processing an error happened. The error stack shows the error detail.

GET_REPLAY_RULES Function

This function returns a list of the replay rules. From the results, you can test if any rule target is replayable with BITAND function.

Syntax

```
DBMS_APP_CONT.GET_REPLAY_RULES (
 replayable IN  BOOLEAN,
 scope      IN  BINARY_INTEGER DEFAULT DBMS_APP_CONT.SCOPE_CURRENT)
 RETURN     BINARY_INTEGER;
```

Parameters

Table 21-5 *GET_REPLAY_RULES Procedure Parameters*

Parameter	Description
replayable	TRUE or FALSE, depending on weather the rule is replayable or not.
scope	Scope of the rules, either dbms_app_cont.scope_current (default) or dbms_app_cont.scope_parent.

RESET_REPLAY_RULES Procedure

This procedure resets all existing replay rules. When you call this procedure, it clears all locally defined rules in the specified scope and restores them to the original state when they were inherited from their parent scope, or the service if parent scope is not specified.

Syntax

```
DBMS_APP_CONT.RESET_REPLAY_RULES (
 targets   IN  BINARY_INTEGER DEFAULT NULL,
 scope     IN  BINARY_INTEGER DEFAULT DBMS_APP_CONT.SCOPE_CURRENT);
```

Parameters

Table 21-6 ***RESET_REPLAY_RULES Procedure Parameters***

Parameter	Description
`targets`	Target effects on which you want to reset the rules.
`scope`	Scope of the rule, either `dbms_app_cont.scope_current` (default) or `dbms_app_cont.scope_parent`.