22

DBMS_APP_CONT_ADMIN

This package provides a collection dba level admin operations in relation to Application Continuity.

This chapter contains the following topics:

- DBMS_APP_CONT_ADMIN Security Model
- Summary of DBMS_APP_CONT_ADMIN Subprograms

DBMS_APP_CONT_ADMIN Security Model

Applications must have the `EXECUTE` privilege on the `DBMS_APP_CONT_ADMIN` package.

Summary of DBMS_APP_CONT_ADMIN Subprograms

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

Table 22-1 DBMS_APP_CONT_ADMIN Package Subprograms

Subprogram	Description
ACCHK_CLEAR_FILTER Procedure	This procedure clears an ACCHK filter by providing a filter type and filter name.
ACCHK_PURGE Procedure	This procedure purges all previously collected ACCHK information.
ACCHK_SET Procedure	This procedure enables or disables data collection for acchk protection for your application when using Application Continuity or Transparent Application Continuity. The `DISABLE_TIME` parameter is used to extend the runtime. Default is `5` minutes.
ACCHK_SET_FILTER Procedure	This procedure sets ACCHK filtering options for `acchk` protection analysis by service name, module name, program name, and machine name.
ACCHK_SHOW_FILTERS Procedure	This procedure uses a cursor to show ACCHK filters ordered by service, program, module, and machine name.
CHECK_REPLAY_RULES Procedure	This procedure shows replay rules for the specified or the current service and returns the replayable targets as a bitmap in targets argument.
ADD_SQL_CONNECTION_TEST Procedure	This procedure adds a new connection test that is used during draining sessions before planned maintenance begins.
DELETE_SQL_CONNECTION_TEST Procedure	This procedure deletes a connection test that is no longer needed for planned draining. Removing a test applies immediately to all RAC instances where the PDB is open.
DISABLE_CONNECTION_TEST Procedure	This procedure disables usage of a connection test during draining of sessions.
DISABLE_FAILOVER Procedure	This procedure disables failover on a given service.
ENABLE_AC Procedure	This procedure enables Application Continuity (AC) on a given service.

Table 22-1 (Cont.) DBMS_APP_CONT_ADMIN Package Subprograms

Subprogram	Description
ENABLE_CONNECTION_TEST Procedure	This procedure enables usage of a connection test for draining database sessions before planned maintenance. Enabling a test applies immediately to all RAC instances where the PDB is open.
ENABLE_RESET_STATE Procedure	This procedure enables clearing the session state usage between requests, so that each new request starts clean (usage web and stateless applications).
ENABLE_TAC Procedure	This procedure enables Transparent Application Continuity (TAC) on a given service.
ENABLE_TAF Procedure	This procedure enables Transparent Application Failover (TAF) for a given service.
ENABLE_TG Procedure	This procedure enables Transaction Guard on a given service.
MODIFY_SERVICE Procedure	This procedure modifies a database service using the given parameters.
RESET_REPLAY_RULES Procedure	This procedure resets replay rules for the specified or the current service.
SET_DRAINING Procedure	This procedure configures draining options for your service, such as timeout value and stop option.
SET_LOAD_BALANCING Procedure	This procedure configures load balancing options for your service.
SET_REPLAY_RULES Procedure	This procedure sets replay rules for the specified or the current service.

ACCHK_CLEAR_FILTER Procedure

This procedure clears an ACCHK filter by providing a filter type, filter name, or all the filters.

- You should call `ACCHK_CLEAR_FILTER` before `ACCHK_SET`, which loads the filters.
- `ACCHK_CLEAR_FILTER` is set at the PDB level.

Syntax

```
DBMS_APP_CONT_ADMIN.ACCHK_CLEAR_FILTER (
   filter_type IN NUMBER   DEFAULT NULL,
   filter_name IN VARCHAR2 DEFAULT NULL,
   purge_all   IN BOOLEAN  DEFAULT FALSE);
```

Parameters

Table 22-2 ACCHK_CLEAR_FILTER Procedure Parameters

Parameter	Description
`filter_type`	Type of the filter, which can be one of the constants `SERVICE_FILTER`, `PROGRAM_FILTER`, `MODULE_FILTER`, or `MACHINE_FILTER` from the `DBMS_APP_CONT_ADMIN` package.
`filter_name`	Name of the filter that you want to delete when `purge_all` is FALSE.

Table 22-2 (Cont.) ACCHK_CLEAR_FILTER Procedure Parameters

Parameter	Description
purge_all	This parameter is used to delete all the filters. • TRUE- enables deletion of all the filters. • FALSE- disables deletion of all the filters. When this parameter is set to FALSE, then only the specified filter is deleted.

Error Messages

Table 22-3 ACCHK_CLEAR_FILTER Procedure Error Messages

Error Code	Description
ORA-20000	Provided filter name exceeds maximum number of characters.
ORA-20000	Invalid filter type value.
ORA-20000	Filter name is not specified.

Examples

The following examples illustrate how to clear an ACCHK service filter:

```
SQL> execute
dbms_app_cont_admin.acchk_clear_filter(DBMS_APP_CONT_ADMIN.SERVICE_FILTER,
'ORACLE.Service1');
```

ACCHK_PURGE Procedure

This procedure purges all previously collected ACCHK information.

- You should call ACCHK_PURGE before ACCHK_SET, which loads the filters.
- ACCHK_PURGE is set at the PDB level.

Syntax

```
DBMS_APP_CONT_ADMIN.ACCHK_PURGE (
   start_time IN DATE    DEFAULT NULL,
   end_time   IN DATE    DEFAULT NULL,
   purge_all  IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 22-4 ACCHK_PURGE Procedure Parameters

Parameter	Description
start_time	The start time from which you want to purge the ACCHK information.
end_time	The end time till when you want to purge the ACCHK information.

Table 22-4 (Cont.) ACCHK_PURGE Procedure Parameters

Parameter	Description
purge_all	This parameter is used to delete all ACCHK information. • TRUE- enables deletion of all ACCHK information. • FALSE- disables deletion of all ACCHK information. When this parameter is set to FALSE, then only the specified ACCHK information is deleted.

Examples

The following examples illustrate how to purge all ACCHK information:

```
SQL> execute dbms_app_cont_admin.acchk_purge(purge_all => TRUE);
```

ACCHK_SET Procedure

This procedure enables or disables data collection for acchk protection reports for your application when using Application Continuity or Transparent Application Continuity. The DISABLE_TIME parameter is used to extend the runtime. Default value is 10 minutes.

Enabling or disabling data collection applies to the level connected, that is, a CDB or a PDB.

- acchk is enabled/disabled at your CDB if connected to the container.
- acchk is enabled/disabled at your PDB only when connected to the PDB.

Data collection applies to new sessions only.

Once enabled, data is collected for the workload run under this service. You can then view this data in the ACCHK_REPORTs and can also be mined in the ACCHK views.

Syntax

```
DBMS_APP_CONT_ADMIN.ACCHK_SET (
   enabled                 IN BOOLEAN,
   disable_time_in_seconds  DEFAULT 600);
```

Parameters

Table 22-5 ACCHK_SET Procedure Parameters

Parameter	Description
enabled	This parameter is used to enable or disable data collection at a CDB or PDB level. • TRUE-enables data collection at this level. • FALSE-explicitly disables data collection.

Table 22-5 (Cont.) ACCHK_SET Procedure Parameters

Parameter	Description
disable_time_in_seconds	Optional parameter used to disable ACCHK tracing automatically in a given number of seconds. The maximum value that you can specify is 3600 seconds. The default value is 600 seconds.

Usage Notes

- This procedure is owned by SYS at CDB$ROOT or PDB level, or by SYS when not multitenant
- The acchk activation is enabled across all instances of RAC supporting that service.
- Enabling is persistent to allow for failover and restart tests, that is, implementation uses ALTER SESSION SET EVENTS ….. SCOPE=BOTH. The enable is per database. For Data Guard, enable and disable must be at each database.

Examples

Application Continuity Protection Check is not enabled by default. Follow this procedure to enable or disable ACCHK and generate reports to check protection level for the applications.

1. Grant read access to the users, who will run the Application Continuity Protection Check report and views, using the ACCHK_READ role:

   ```
   GRANT ACCHK_READ TO USER;
   ```

2. Enable Application Continuity tracing for your applications using the dbms_app_cont_admin.acchk_set(true) procedure:

   ```
   SQL> execute dbms_app_cont_admin.acchk_set(true);
   ```

   By default, ACCHK is disabled automatically after 600 seconds. You can specify a lower number to reduce the auto disable time. For example, to disable ACCHK after 300 seconds:

   ```
   SQL> dbms_app_cont_admin.acchk_set(true,300);
   ```

   The dbms_app_cont_admin.acchk_set(true) procedure enables Application Continuity tracing at the database level to which you are connected. If you are connected at the CDB level, then tracing is enabled for the CDB, and if you are connected at the PDB level, then tracing is enabled for the PDB.

   > **Note:**
   >
   > Set the COMPATIBLE parameter to 12.2.0 or greater.

3. To disable Application Continuity tracing for new sessions in your applications:

   ```
   SQL> execute dbms_app_cont_admin.acchk_set(false);
   ```

> **Note:**
>
> The tracing will not be disabled for the current sessions until the sessions are terminated.

ACCHK_SET_FILTER Procedure

This procedure sets ACCHK filtering options for `acchk` protection analysis by service name, module name, program name, and machine name.

Filter options are cumulative by multiple calls of this procedure.

- You should call `ACCHK_SET_FILTER` before `ACCHK_SET`, which loads the filters.
- `ACCHK_SET_FILTER` is set at the PDB level.

Syntax

```
DBMS_APP_CONT_ADMIN.ACCHK_SET_FILTER (
   filter_type IN NUMBER,
   filter_name IN VARCHAR2);
```

Parameters

Table 22-6 ACCHK_SET_FILTER Procedure Parameters

Parameter	Description
`filter_type`	Type of the filter, which can be one of the constants`SERVICE_FILTER`, `PROGRAM_FILTER`, `MODULE_FILTER`, or `MACHINE_FILTER` from the `DBMS_APP_CONT_ADMIN` package.
`filter_name`	Name of the filter, which must match a service name, program name, module name, or machine name.

Error Messages

Table 22-7 ACCHK_SET_FILTER Procedure Error Messages

Error Code	Description
`ORA-20000`	Provided filter name exceeds maximum number of characters.
`ORA-20000`	Service `service_name` does not exist.
`ORA-20000`	Invalid filter type value.
`ORA-20000`	Filter name is not specified.
`ORA-20000`	The limit has been reached, only 1024 filters are allowed.
`ORA-20000`	Filter name already exists.

Examples

The following examples illustrate how to set filters for ACCHK constants:

- To set a service filter:

```
SQL> execute
dbms_app_cont_admin.acchk_set_filter(DBMS_APP_CONT_ADMIN.SERVICE_FILTER,
'ORACLE.Service1');
```

- To set a program filter:

```
SQL> execute
dbms_app_cont_admin.acchk_set_filter(DBMS_APP_CONT_ADMIN.PROGRAM_FILTER,
'Oracle.Program');
```

- To set a module filter:

```
SQL> execute
dbms_app_cont_admin.acchk_set_filter(DBMS_APP_CONT_ADMIN.MODULE_FILTER,
'Oracle.Module');
```

ACCHK_SHOW_FILTERS Procedure

This procedure uses a cursor to show ACCHK filters ordered by service, program, module, and machine name.

Syntax

```
DBMS_APP_CONT_ADMIN.ACCHK_SHOW_FILTERS;
```

Examples

The following examples illustrate how to show ACCHK filters:

```
SQL> execute dbms_app_cont_admin.acchk_show_filters;
ResultSet TYPE  FILTER_NAME  SERVICE          PROGRAM            MODULE
#1                           oracle.service1  oracle.program     oracle.module
```

ADD_SQL_CONNECTION_TEST Procedure

This procedure adds a new connection test that is used during draining sessions before planned maintenance begins. Use this procedure when the SQL connection test is not covered by standard tests. The test is enabled when added. If the optional service name qualifier is provided, the test only applies only to that service name.

Syntax

```
DBMS_APP_CONT_ADMIN.ADD_SQL_CONNECTION_TEST (
   connection_test          IN VARCHAR2
   service_name             IN VARCHAR2   DEFAULT NULL);
```

Parameters

Table 22-8 ADD_SQL_CONNECTION_TEST Procedure Parameters

Parameter	Description
CONNECTION_TEST	The SQL text used to test and drain connections.

Table 22-8 (Cont.) ADD_SQL_CONNECTION_TEST Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier.

Usage Notes

The ADD_SQL_CONNECTION_TEST Procedure adds a connection test for the purpose of draining sessions before planned maintenance begins. The connection test is used by the application to test connections that are marked for draining. Sessions are set for draining at stop and relocate operations for services or PDBs. When set the RDBMS closes the connection while draining so the application sees no errors during planned maintenance. You can enter as many CONNECTION TESTs as needed. They are used only during planned maintenance. The tests apply to all RAC instances.

Check online documentation for latest updates on service qualifier availability.

Added connection can be viewed by querying the view DBA_CONNECTION_TESTS.

This procedure is owned by SYS and is granted to users for execution at CDB$ROOT or PDB levels, or when not multitenant, at dictionary level.

CHECK_REPLAY_RULES Procedure

This procedure shows replay rules for the specified or the current service and returns the replayable targets as a bitmap in targets argument.

Syntax

```
DBMS_APP_CONT_ADMIN.CHECK_REPLAY_RULES (
 service_name  IN  VARCHAR2        DEFAULT NULL
 targets       OUT BINARY_INTEGER);
```

Parameters

Table 22-9 *CHECK_REPLAY_RULES Procedure Parameters*

Parameter	Description
service_name	Optional service name qualifier. If the optional SERVICE_NAME qualifier is not provided, then replay rules are displayed for the current service.
targets	The replayable targets in a bitmap. You can use BITAND function to test if any one of these targets is replayable: • dbms_app_cont_admin.side_effects • dbms_app_cont_admin.autonomous_transactions • dbms_app_cont_admin.database_links

CREATE_CHILD_SERVICE Procedure

This procedure creates a child service for an existing parent service.

> **Note:**
>
> If the parent service is started, then the child service will be started on all of the same instances on which the parent service is active.

Syntax

```
DBMS_APP_CONT_ADMIN.CREATE_CHILD_SERVICE (
 child_service_name  IN  VARCHAR2,
 parent_service_name IN  VARCHAR2);
```

Parameters

Table 22-10 *CREATE_CHILD_SERVICE Procedure Parameters*

Parameter	Description
child_service_name	Name of the new child service that you want to create.
parent_service_name	Name of the parent service for which you want to create a child service.

DELETE_CHILD_SERVICE Procedure

This procedure deletes a child service from an existing parent service.

> **Note:**
>
> If the child service is active on any instance, then the child service will be stopped before it is deleted.

Syntax

```
DBMS_APP_CONT_ADMIN.DELETE_CHILD_SERVICE (
 child_service_name  IN  VARCHAR2,
 parent_service_name IN  VARCHAR2);
```

Parameters

Table 22-11 *DELETE_CHILD_SERVICE Procedure Parameters*

Parameter	Description
child_service_name	Name of the existing child service that you want to delete.
parent_service_name	Name of the parent service from which you want to delete the specified child service.

DELETE_SQL_CONNECTION_TEST Procedure

This procedure deletes a connection test that is no longer needed for planned draining. Removing a test applies immediately to all RAC instances where the PDB is open.

Syntax

```
DBMS_APP_CONT_ADMIN.DELETE_SQL_CONNECTION_TEST (
   connection_test          IN VARCHAR2
   service_name             IN VARCHAR2   DEFAULT NULL);
```

Parameters

Table 22-12 DELETE_SQL_CONNECTION_TEST Procedure Parameters

Parameter	Description
CONNECTION_TEST	The SQL text used to test and drain connections.
SERVICE_NAME	Optional service name qualifier. If the optional SERVICE_NAME qualifier is provided, only the test for that service name is deleted.

Usage Notes

If you are not certain if a test should be deleted, you can disable the test using DISABLE_CONNECTION_TEST Procedure. Only custom SQL tests can be deleted. Predefined tests cannot be deleted. Check for latest updates on service qualifier availability.

This procedure is owned by SYS at CDB$ROOT or PDB level, or SYS for when not multitenant.

Connection tests and their status can be checked by querying the view DBA_CONNECTION_TESTS.

DISABLE_CONNECTION_TEST Procedure

This procedure disables usage of a connection test during draining of sessions. Disabling a test applies immediately to all RAC instances where the PDB is open.

Syntax

```
DBMS_APP_CONT_ADMIN.DISABLE_CONNECTION_TEST (
   connection_test_type     IN VARCHAR2,
   connection_test          IN VARCHAR2,
   service_name             IN VARCHAR2   DEFAULT NULL);
```

Parameters

Table 22-13 DISABLE_CONNECTION_TEST Procedure Parameters

Parameter	Description
CONNECTION_TEST_TYPE	The permitted values are: • DBMS_APP_CONT_ADMIN.SQL_TEST • DBMS_APP_CONT_ADMIN.PING_TEST • DBMS_APP_CONT_ADMIN.ENDREQUEST_TEST

Table 22-13 (Cont.) DISABLE_CONNECTION_TEST Procedure Parameters

Parameter	Description
CONNECTION_TEST	The SQL text used to test and drain connections. This parameter is allowed only if the value of CONNECTION_TEST_TYPE is SQL_TEST.
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, only the test for that service name is enabled. A disable at service name level takes precedence over an enable at PDB level. That is the PDB can be enabled, and the service disabled.

Usage Notes

This procedure is owned by SYS and is granted to users for execution at CDB$ROOT or PDB levels, or when not multitenant, at dictionary level.

Connection tests and their status can be checked by querying the view DBA_CONNECTION_TESTS.

DISABLE_FAILOVER Procedure

This procedure disables failover on a given service.

Syntax

```
DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER (
    service_name IN VARCHAR2);
```

Parameters

Table 22-14 DISABLE_FAILOVER Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, then failover is disabled only for that service name.

Usage Notes

- You must have the PDBADMIN user permissions to use this procedure.
- Use the full service name on which you want to disable the failover.

Examples

The following example illustrates how to disable failover for a service:

```
SQL> execute dbms_app_cont_admin.disable_failover('TPURGENT');
```

DISABLE_SMART_CONNECTION Procedure

This procedure disables smart connection for the specified service, that is, `GOAL` is set to `GOAL_NONE`.

> **Note:**
>
> If the service has children, then smart connection will be disabled for these services as well.

Syntax

```
DBMS_APP_CONT_ADMIN.DISABLE_SMART_CONNECTION (
 service_name  IN  VARCHAR2);
```

Parameters

Table 22-15 *DISABLE_SMART_CONNECTION Procedure Parameters*

Parameter	Description
`service_name`	Optional service name qualifier. If the optional `SERVICE_NAME` qualifier is not provided, then smart connection is disabled on the current service.

ENABLE_AC Procedure

This procedure enables Application Continuity (AC) on a given service.

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_AC (
   service_name              IN VARCHAR2,
   failover_restore          IN VARCHAR2       DEFAULT 'LEVEL1'
   replay_initiation_timeout IN BINARY_INTEGER DEFAULT 300);
```

Parameters

Table 22-16 ENABLE_AC Procedure Parameters

Parameter	Description
`SERVICE_NAME`	Optional service name qualifier. If the optional service name qualifier is provided, AC is enabled only for the specified service. An enable at service name level overrides any higher-level disables. That is, the PDB can be disabled, and the service enabled.
`FAILOVER_RESTORE`	Failover restore. Possible values are `NONE` or `LEVEL1`.
`REPLAY_INITIATION_TIMEOUT`	Replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

Usage Notes

- You must have the `PDBADMIN` user permissions to use this procedure.
- Use the full service name on which you want to enable AC.

Examples

The following example illustrates how to enable Application Continuity for your service:

```
SQL> execute dbms_app_cont_admin.enable_ac('TPURGENT', 'LEVEL1', 600);
```

ENABLE_CONNECTION_TEST Procedure

This procedure enables usage of a connection test for draining database sessions before planned maintenance. Enabling a test applies immediately to all RAC instances where the PDB is open.

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_CONNECTION_TEST (
   connection_test_type     IN VARCHAR2,
   connection_test          IN VARCHAR2,
   service_name             IN VARCHAR2   DEFAULT NULL);
```

Parameters

Table 22-17 ENABLE_CONNECTION_TEST Procedure Parameters

Parameter	Description
`CONNECTION_TEST_TYPE`	The connection type used when managing connection tests for draining before planned maintenance. See ADD, DELETE, ENABLE, DISABLE procedures for connection tests. The permitted values are: • `DBMS_APP_CONT_ADMIN.SQL_TEST` • `DBMS_APP_CONT_ADMIN.PING_TEST` • `DBMS_APP_CONT_ADMIN.ENDREQUEST_TEST` • `DBMS_APP_CONT_ADMIN.BEGINREQUEST_TEST`
`CONNECTION_TEST`	The SQL text used to test and drain connections at the RDBMS before planned maintenance starts. This parameter is allowed only if the value of `CONNECTION_TEST_TYPE` is `SQL_TEST`.
`SERVICE_NAME`	Optional service name qualifier. If the optional service name qualifier is provided, only the test for that service name is enabled. An enable at service name level overrides any higher-level disables. That is, the PDB can be disabled, and the service enabled.

Usage Notes

- This procedure is owned by SYS and is granted to users for execution at CDB$ROOT or PDB levels, or when not multitenant, at dictionary level
- ENABLE_CONNECTION_TEST enables a connection test for draining sessions during planned maintenance. The enable operation applies to all RAC instances where the PDB is open. It persists across database restarts.
- This procedure is owned by SYS and is granted to users for execution at CDB$ROOT or PDB levels, or when not multitenant, at dictionary level.

ENABLE_RESET_STATE Procedure

This procedure enables clearing the session state usage between requests, so that each new request starts clean (usage web and stateless applications).

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_RESET_STATE (
   service_name IN VARCHAR2,
   level        IN VARCHAR2 DEFAULT AUTO);
```

Parameters

Table 22-18 ENABLE_RESET_STATE Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, RESET_STATE is enabled only for the specified service.
LEVEL	Level of the Oracle Database reset session state configuration. Available options are LEVEL1, LEVEL2, or AUTO.

Usage Notes

- You must have the PDBADMIN user permissions to use this procedure.
- Use the full service name for which you want to clear the session state.

Examples

The following example illustrates how to enable reset state for your service:

```
SQL> execute dbms_app_cont_admin.enable_reset_state('TPURGENT', 'AUTO');
```

ENABLE_SMART_CONNECTION Procedure

This procedure enables smart connection for the specified service, that is, `GOAL` is set to `SMART_CONN`.

> **Note:**
>
> If the service has children, then smart connection will be enabled for these services as well.

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_SMART_CONNECTION (
 service_name  IN  VARCHAR2);
```

Parameters

Table 22-19 *ENABLE_SMART_CONNECTION Procedure Parameters*

Parameter	Description
`service_name`	Optional service name qualifier. If the optional `SERVICE_NAME` qualifier is not provided, then smart connection is enabled for the current service.

ENABLE_TAC Procedure

This procedure enables Transparent Application Continuity (TAC) on a given service.

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_TAC (
   service_name              IN VARCHAR2,
   failover_restore          IN VARCHAR2       DEFAULT 'AUTO'
   replay_initiation_timeout IN BINARY_INTEGER DEFAULT 300
   session_state_consistency IN VARCHAR2       DEFAULT 'AUTO');
```

Parameters

Table 22-20 ENABLE_TAC Procedure Parameters

Parameter	Description
`SERVICE_NAME`	Optional service name qualifier. If the optional service name qualifier is provided, TAC is enabled only for the specified service. An enable at service name level overrides any higher-level disables. That is, the PDB can be disabled, and the service enabled.
`FAILOVER_RESTORE`	Failover restore. Possible values are `AUTO` or `LEVEL1`.

Table 22-20 (Cont.) ENABLE_TAC Procedure Parameters

Parameter	Description
REPLAY_INITIATION_TIMEOUT	Replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.
SESSION_STATE_CONSISTENCY	Session State Consistency. Possible values are AUTO or HYBRID.

Exceptions

- When FAILOVER_RESTORE is set to NULL or FAILOVER_RESTORE is not set to AUTO or LEVEL1: ORA-20000 - Invalid failover_restore parameter.
- When SESSION_STATE_CONSISTENCY is set to NULL or SESSION_STATE_CONSISTENCY is not set as AUTO: ORA-20000 - Invalid session_state_consistency parameter.

Usage Notes

- You must have the PDBADMIN user permissions to use this procedure.
- Use the full service name on which you want to enable TAC.

Examples

The following example illustrates how to enable Transparent Application Continuity for your service:

```
SQL> execute dbms_app_cont_admin.enable_tac('TPURGENT', 'AUTO', 600, 'AUTO');
```

ENABLE_TAF Procedure

This procedure enables Transparent Application Failover (TAF) for a given service.

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_TAF (
   service_name      IN VARCHAR2,
   failover_type     IN VARCHAR2  DEFAULT 'AUTO'
   failover_restore  IN VARCHAR2  DEFAULT 'LEVEL1');
```

Parameters

Table 22-21 ENABLE_TAF Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, TAF is enabled only for the specified service. An enable at service name level overrides any higher-level disables. That is, the PDB can be disabled, and the service enabled.
FAILOVER_TYPE	Failover type. Possible values are LOW, MEDIUM, or HIGH.

Table 22-21 (Cont.) ENABLE_TAF Procedure Parameters

Parameter	Description
FAILOVER_RESTORE	Failover restore. Possible values are NONE or LEVEL1.

Usage Notes

- You must have the PDBADMIN user permissions to use this procedure.
- Use the full service name on which you want to enable TAF.

Examples

The following example illustrates how to enable TAF SELECT for the current service:

```
SQL> execute dbms_app_cont_admin.enable_taf(`LOW`);
```

The following example illustrates how to enable TAF BASIC for the current service:

```
SQL> execute dbms_app_cont_admin.enable_taf(`MEDIUM`, `SESSION`);
```

ENABLE_TG Procedure

This procedure enables Transaction Guard on a given service.

Syntax

```
DBMS_APP_CONT_ADMIN.ENABLE_TG (
   service_name IN VARCHAR2);
```

Parameters

Table 22-22 ENABLE_TG Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, AC is enabled only for the specified service. An enable at service name level overrides any higher-level disables. That is, the PDB can be disabled, and the service enabled.

Usage Notes

- You must have the PDBADMIN user permissions to use this procedure.
- Use the full service name on which you want to enable Transaction Guard.

Examples

The following example illustrates how to enable Transaction Guard for your service:

```
SQL> execute dbms_app_cont_admin.enable_tg('TPURGENT');
```

MODIFY_SERVICE Procedure

This procedure modifies a database service using the given parameters.

Syntax

```
DBMS_APP_CONT_ADMIN.MODIFY_SERVICE(
   service_name   IN VARCHAR2,
   service_params IN svc_parameter_array);
```

Parameters

Table 22-23 MODIFY_SERVICE Procedure Parameters

Parameter	Description
service_name	Name of the service, limited to 64 characters in the Data Dictionary
parameter_array	Associative array with name/value pairs of the service attributes. Supported names: • aq_ha_notifications • auto_connection_rebalance • clb_goal • commit_fast_path • commit_outcome • drain_timeout • edition • failover_delay • failover_method • failover_restore • failover_retries • failover_type • goal • replay_initiation_timeout • reset_state • retention_timeout • session_state_consistency • sql_translation_profile • stop_option • template_timeout • true_cache_service
aq_ha_notifications	Determines whether Fast Application Notification (FAN) is enabled for OCI/OCCI/ODP. In Oracle Database12*c*, FAN uses Oracle Notification Services (ONS). This parameter is still used to enable FAN. FAN is recommended for all High Availability systems, and is on by default for Application Continuity

Table 22-23 (Cont.) MODIFY_SERVICE Procedure Parameters

Parameter	Description
auto_connection_rebalance	Automatically balances the workload management.
clb_goal	Connection Load Balancing goal, either LONG or SHORT.
commit_fast_path	Enable or disable Oracle Database transaction guard.
commit_outcome	Determines whether transaction COMMIT outcome is accessible after the COMMIT has executed. While the database guarantees that COMMIT is durable, this ensures that the outcome of the COMMIT is durable.Applications use the feature to probe the status of the commit last executed after an outage, and is available to applications to determine an outcome. Note: • Invoking the GET_LTXID_OUTCOME Procedure of the DBMS_APP_CONT package requires that the commit_outcome attribute be set. • commit_outcome has no effect on active Data Guard and read-only databases. • commit_outcome is allowed only on user-defined database services
drain_timeout	When this parameter is set, all sessions connected to that service are drained by the client drivers and pools using Fast Connection Failover (FCF). The drain_timeout can be set on the service, to stop and relocate drains for this time by default.
edition	If this argument has a non-NULL value, this provides the initial session edition for subsequent database connections using this service that do not specify an edition. If no value is specified, this argument has no effect. During service creation or modification, no validation is performed on this parameter. At connection time, if the connecting user does not have USE privilege on the edition, or the edition does not exist, this raises the error ORA-38802 (edition does not exist).
failover_delay	Delay in seconds between connection retries for Application Continuity and TAF. The default is 10 seconds for Application Continuity. Do not use a 0-second delay if the service needs time to failover and register. Long delays are good for planned outages and to failover to Data Guard. Short delays work well with Oracle RAC when the service is already available.
failover_method	Failover TYPE for the service for Application Continuity and TAF. If the failover_type is set to TRANSACTION on the service, this automatically sets COMMIT_OUTCOME to TRUE. JDBC Replay Driver uses the FAILOVER_TYPE service attribute setting of TRANSACTION for TRANSACTION failover. OCI uses the older settings of SELECT and SESSION. The server only accepts FAILOVER_METHOD = BASIC with the TRANSACTION setting.
failover_restore	For Application Continuity, when the failover_restore parameter is set, the session states are restored before replaying for ODP.NET and Java. Use LEVEL1 for ODP.NET and Java with Application Continuity to restore the initial state. For AC OCI, use NONE for applications that are not STATIC.

Table 22-23 (Cont.) MODIFY_SERVICE Procedure Parameters

Parameter	Description
failover_retries	Number of connection retries for Application Continuity and TAF. Using the failover_retries and failover_delay parameters, the failover can be delayed until the service is next available. This parameter is for connecting. It does not control the number of failovers, which is 3 for each incident for Application Continuity.
failover_type	Failover TYPE for the service for Application Continuity and TAF.
goal	Workload management goal directive for the service. Valid values: • DBMS_SERVICE.GOAL_SERVICE_TIME • DBMS_SERVICE.GOAL_THROUGHPUT • DBMS_SERVICE.GOAL_NONE
replay_initiation_timeout	For Application Continuity, replay_initiation_timeout is the difference between the time of original execution of first operation of a request, and the time that the replay is ready to start after a successful reconnect. Replay initiation time is measured from the time that the request was originally submitted until the time that replay has connected and is ready to replay. When replay is expected, keep this value high. Default is 900 seconds.
reset_state	Use to clean session state automatically by Oracle Database between requests.
retention_timeout	Used in conjunction with commit_outcome, it determines the amount of time (in seconds) that the COMMIT_OUTCOME is retained. Default is 24 hours (86400). Maximum value is 30 days (2592000).
session_state_consistency	Describes how nontransactional is changed during a request. This parameter is considered only if failover_type is set to TRANSACTION for Application Continuity. Examples of session state are NLS settings, optimizer preferences, event settings, PL/SQL global variables, temporary tables, advanced queues, LOBs, and result cache. If these values change after the request starts, set to DYNAMIC (default). Almost all applications should use DYNAMIC mode. If you are unsure, use DYNAMIC mode.
sql_translation_profile	Name of SQL translation profile.
stop_option	Stop options for the service.
template_timeout	Template timeout time in seconds.
true_cache_service	Name of the True Cache service being registered with the primary service.

Examples

```
DECLARE
params dbms_app_cont_admin.svc_parameter_array;
BEGIN
   params('FAILOVER_TYPE')             :='TRANSACTION';
   params('FAILOVER_RESTORE')          :='LEVEL2';
   params('RESET_STATE')               :='NONE';
   params('SESSION_STATE_CONSISTENCY') :='DYNAMIC';
   DBMS_APP_CONT_ADMIN.MODIFY_SERVICE('ernie.example.com', params);
END;
```

RESET_REPLAY_RULES Procedure

This procedure resets replay rules for the specified or the current service.

> **Note:**
>
> - When resetting an AC service (`FAILOVER_TYPE = TRANSACTION`), if targets is not specified, then all targets become replayable, meaning allowing all side effects to be replayed (default).
> - When resetting a TAC service (`FAILOVER_TYPE = AUTO`), if targets is not specified, all targets become un-replayable, meaning not allowing any side effect to be replayed (default).

Syntax

```
DBMS_APP_CONT_ADMIN.RESET_REPLAY_RULES (
 service_name  IN  VARCHAR2        DEFAULT NULL
 targets       IN  BINARY_INTEGER  DEFAULT NULL);
```

Parameters

Table 22-24 *RESET_REPLAY_RULES Procedure Parameters*

Parameter	Description
`service_name`	Optional service name qualifier. If the optional `SERVICE_NAME` qualifier is not provided, then replay rules are reset for the current service.
`targets`	Targets for the replay rule. The permitted values are: • `dbms_app_cont_admin.side_effects` • `dbms_app_cont_admin.autonomous_transactions` • `dbms_app_cont_admin.database_links` If multiple targets are required, then pass them as `bit or` of individual targets, such as: `dbms_app_cont.side_effects + dbms_app_cont.autonomous_transactions`

SET_DRAINING Procedure

This procedure configures draining options for your service, such as timeout value and stop option.

Syntax

```
DBMS_APP_CONT_ADMIN.SET_DRAINING (
   service_name  IN VARCHAR2,
   drain_timeout IN BINARY_INTEGER DEFAULT 300
   stop_option   IN VARCHAR2       DEFAULT 'NONE');
```

Parameters

Table 22-25 SET_DRAINING Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, draining is set only for that service.
DRAIN_TIMEOUT	Specify the time, in seconds, allowed for resource draining to be completed. Accepted values are an empty string (""), 0, or any positive integer.
STOP_OPTION	Specify the method of stopping the service. Available options are NONE, IMMEDIATE, or TRANSACTIONAL.

Examples

The following example illustrates how to set draining options for your service:

```
SQL> execute dbms_app_cont_admin.set_draining('TPURGENT', 300, 'IMMEDIATE');
```

SET_LOAD_BALANCING Procedure

This procedure configures load balancing options for your service.

Syntax

```
DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING (
   service_name IN VARCHAR2,
   goal         IN VARCHAR2);
```

Parameters

Table 22-26 SET_LOAD_BALANCING Procedure Parameters

Parameter	Description
SERVICE_NAME	Optional service name qualifier. If the optional service name qualifier is provided, load balancing is set only for the specified service.
GOAL	Load balancing goal. Possible values are: • CLBGOAL- Connection Load Balancing Goal. • RLBGOAL- Runtime Load Balancing Goal.

Examples

The following example illustrates how to set load balancing goal for your service:

```
SQL> execute dbms_app_cont_admin.set_load_balancing('TPURGENT', 'CLBGOAL');
```

SET_REPLAY_RULES Procedure

This procedure sets replay rules for the specified or the current service.

The replay rules are used by Application Continuity to determine whether side effects can be replayed or not.

Syntax

```
DBMS_APP_CONT_ADMIN.SET_REPLAY_RULES (
 service_name  IN  VARCHAR2        DEFAULT NULL
 replayable    IN  BOOLEAN,
 targets       IN  BINARY_INTEGER);
```

Parameters

Table 22-27 *SET_REPLAY_RULES Procedure Parameters*

Parameter	Description
`service_name`	Optional service name qualifier. If the optional `SERVICE_NAME` qualifier is not provided, then replay rules are set for the current service.
`replayable`	`TRUE` or `FALSE`, depending on whether the rule is replayable or not.
`targets`	Target effects on which this rule is applied. The permitted values are: • `dbms_app_cont_admin.side_effects` • `dbms_app_cont_admin.autonomous_transactions` • `dbms_app_cont_admin.database_links` If multiple targets are required, then pass them as `bit or` of individual targets, such as: `dbms_app_cont.side_effects + dbms_app_cont.autonomous_transactions`