214
DBMS_USERDIAG

The DBMS_USERDIAG package allows you to perform a limited set of diagnosis operations on the PDB.

This chapter contains the following topics:

- DBMS_USERDIAG Overview
- Summary of DBMS_USERDIAG Subprograms

Enabling traces on cloud requires additional configuration as documented here: Perform SQL Tracing on Autonomous Database.

DBMS_USERDIAG Overview

The DBMS_USERDIAG package is for diagnosis and allows you to set up a trace within a PDB.

The DBMS_USERDIAG package utilizes underneath DBMS_SYSTEM functionality but with a narrow set of definitions which restrict arbitrary event settings.

For a given PDB, the DBMS_USERDIAG package allows you:

- to enable the SQL trace at a given level.
- to disable the SQL trace.
- to check the SQL trace.

Most of the regular diagnostic mechanisms have been restricted outside of a given PDB using lockdown profiles, so that arbitrary events cannot be enabled from user sessions in a shared tenancy in CBD deployments in cloud instances. In particular, "alter session set events" statement is blocked in cloud deployments, because it can be misused to set events/actions which may change code-path execution or simulate errors.

Summary of DBMS_USERDIAG Subprograms

The DBMS_USERDIAG package uses ENABLE_SQL_TRACE_EVENT and CHECK_SQL_TRACE_EVENT subprograms to enable, disable, and monitor sql_trace events.

Table 214-1 DBMS_USERDIAG Package Subprograms

Subprogram	Description
CHECK_SQL_TRACE_EVENT Procedure	Checks the current sql_trace event and retrieves the level.
ENABLE_SQL_TRACE_EVENT Procedure	Enables sql_trace event at a given level in user session, and generates SQL traces into respective process trace files.
GET_CALL_ERROR_MSG Function	Obtains the error message if the last call to DBMS_USERDIAG API returned an error.
GET_CALL_STATUS Function	Obtains the status of the last call to the DBMS_USERDIAG API.

Table 214-1 (Cont.) DBMS_USERDIAG Package Subprograms

Subprogram	Description
SET_EXCEPTION_MODE Procedure	Raises an exception on any error or silently ignores the same (default).
SET_TRACEFILE_IDENTIFIER Procedure	Sets a custom trace file identifier for the active trace file in the current ADR home.
TRACE Procedure	Traces message to the trace file or alert log.

CHECK_SQL_TRACE_EVENT Procedure

This procedure can be used by the current user to check the current `sql_trace` event and retrieves the level.

Syntax

```
DBMS_USERDIAG.CHECK_SQL_TRACE_EVENT(
   sql_id                IN VARCHAR2       DEFAULT NULL
   sys                   IN BINARY_INTEGER DEFAULT NULL
   level                 OUT BINARY_INTEGER);
```

Parameters

Table 214-2 CHECK_SQL_TRACE_EVENT Procedure Parameters

Parameter	Description
`level`	Required. Output of the current level at which `sql_trace` event is enabled.
`sql_id`	`sql_trace` in scope of a given `sql_id` alone. This is currently ignored.
`sys`	Optional. Set event instance-wide or PDB-wide. Default 0. This is currently ignored.

ENABLE_SQL_TRACE_EVENT Procedure

This procedure enables `sql_trace` event (`sql_trace/10046`) at a given `level` in user session, and generates SQL traces into respective process trace files.

Optionally a `sql_id` can also be specified in which case tracing is enabled for that `sql_id` alone. Default is to enable events without sql scope. Default is to set event in current session alone. Event can be disabled by setting `disable` to non-zero value (default zero).

Syntax

```
DBMS_USERDIAG.ENABLE_SQL_TRACE_EVENT(
   level                 IN  BINARY_INTEGER DEFAULT 1,
   sid                   IN  BINARY_INTEGER DEFAULT 0,
   ser                   IN  BINARY_INTEGER DEFAULT 0,
   binds                 IN  BINARY_INTEGER DEFAULT 0,
   waits                 IN  BINARY_INTEGER DEFAULT 0,
   plan_stat             IN  VARCHAR2       DEFAULT NULL,
   sql_id                IN  VARCHAR2       DEFAULT NULL,
```

```
   disable                IN  BINARY_INTEGER DEFAULT 0,
   sys                    IN  BINARY_INTEGER DEFAULT 0);
```

Parameters

Table 214-3 ENABLE_SQL_TRACE_EVENT Procedure Parameters

Parameter	Description
level	Optional. Specifies the level associated with sql_trace event. Other settings augment the level value. For example, binds=>true adds to whatever level value has been provided by the user. This is kept for ease of usage.
sid	Optional. Target session identifier. sid, ser pair must be validated to be within current PDB id.
ser	Optional. Target serial number.
binds	Optional. A non-zero value traces bind values. Maps to level 4.
waits	Optional. A non-zero value traces waits. Maps to level 8.
plan_stat	Optional. Allowed values: FIRST_EXECUTION (default), NEVER, ALL_EXECUTIONS, and ADAPTIVE
sql_id	Enable sql_trace in scope of a given sql_id alone
disable	Optional. Non-zero value disables the already set sql_trace event.
sys	Optional. Set event instance-wide or PDB-wide. Default 0.

GET_CALL_ERROR_MSG Function

This function is used to obtain the error message if the last call to DBMS_USERDIAG API returned an error.

Syntax

```
DBMS_USERDIAG.GET_CALL_ERROR_MSG RETURN VARCHAR2;
```

Return Value

If the previous call to DBMS_USERDIAG was unsuccessful, the VARCHAR2 contains the error message associated with it.

If the previous call to DBMS_USERDIAG was successful, the value returned is NULL.

GET_CALL_STATUS Function

This function is used to obtain the status of the last call to the DBMS_USERDIAG API.

Syntax

```
DBMS_USERDIAG.GET_CALL_STATUS RETURN NUMBER;
```

Return Value

Returns a number for the status of the last call to the DBMS_USERDIAG API.

If the previous call was successful, the value of this call is `NOERROR(0)`.

If the previous call was unsuccessful, the value of this call is the error code number.

SET_EXCEPTION_MODE Procedure

This procedure raises an exception on any error or silently ignores the same (default).

Syntax

```
DBMS_USERDIAG.SET_EXCEPTION_MODE(
   exc_mode IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 214-4 SET_EXCEPTION_MODE Procedure Parameters

Parameter	Description
`exc_mode`	When `TRUE`, an exception is raised on any error. When `FALSE` (default), errors are ignored.

SET_TRACEFILE_IDENTIFIER Procedure

This procedure is used to set a custom trace file identifier for the active trace file in the current ADR home.

Syntax

```
DBMS_USERDIAG.SET_TRACEFILE_IDENTIFIER(
   trc_identifier IN VARCHAR2);
```

Parameters

Table 214-5 SET_TRACEFILE_IDENTIFIER Procedure Parameters

Parameter	Description
`trc_identifier`	Specifies the trace identifier for the active trace file in the current ADR home.

TRACE Procedure

This procedure is used to trace message to the trace file or alert log.

Syntax

```
DBMS_USERDIAG.TRACE(
   message IN VARCHAR2,
   alert   IN BOOLEAN   DEFAULT FALSE);
```

Parameters

Table 214-6 TRACE Procedure Parameters

Parameter	Description
`message`	Specifies the text or message to be traced.
`alert`	When `TRUE`, the alert logs are included with the trace file for the trace operation. When `FALSE` (default), the alert logs are ignored and only the trace file is used for the trace operation.