# 204
# DBMS_TRACE

The DBMS_TRACE package contains the interface to trace PL/SQL functions, procedures, and exceptions.

This chapter contains the following topics:



## DBMS_TRACE Overview

DBMS_TRACE provides subprograms to start and stop PL/SQL tracing in a session. Oracle collects the trace data as the program executes and writes it to database tables.

A typical session involves:

1. (Optional) Limit tracing to specific subprograms and choose a tracing level.

   Tracing all subprograms and exceptions in a large program can produce huge amounts of data that are difficult to manage.
2. Starting PL/SQL tracing in session (DBMS_TRACE.SET_PLSQL_TRACE).
3. Running an application to be traced.
4. Stopping PL/SQL tracing in session (DBMS_TRACE.CLEAR_PLSQL_TRACE).

After you have collected data with Trace, you can query the database tables that contain the performance data and analyze it in the same way that you analyze the performance data from Profiler.

## DBMS_TRACE Security Model

This package must be created under SYS.

## DBMS_TRACE Constants

DBMS_TRACE defines constants to use when specifying parameter values.

These constants are shown in the following table.

**Table 204-1 DBMS_TRACE Event Constants**

| Name | Type | Value | Description |
|---|---|---|---|
| TRACE_ALL_CALLS | INTEGER | 1 | Traces calls or returns |
| TRACE_ENABLED_CALLS | INTEGER | 2 | |
| TRACE_ALL_EXCEPTIONS | INTEGER | 4 | Traces exceptions |
| TRACE_ENABLED_EXCEPTIONS | INTEGER | 8 | Traces exceptions and handlers |
| TRACE_LIMIT | INTEGER | 16 | Save only the last few records. This allows tracing up to a problem area, without filling the database up with masses of irrelevant information. If event 10940 is set, the limit is 1023*(the value of event 10940). This can be overridden by the use of "TRACE_LIMIT" flag. |
| TRACE_ALL_SQL | INTEGER | 32 | Traces SQL statements |
| TRACE_ENABLED_SQL | INTEGER | 64 | Traces SQL statements at PL/SQL level. This does not invoke SQL Trace |
| TRACE_ALL_LINES | INTEGER | 128 | Traces each line |
| TRACE_ENABLED_LINES | INTEGER | 256 | |
| TRACE_PAUSE | INTEGER | 4096 | Pauses tracing |
| TRACE_RESUME | INTEGER | 8192 | Resume tracing |
| TRACE_STOP | INTEGER | 16384 | Stops tracing |
| NO_TRACE_ADMINISTRATIVE | INTEGER | 32768 | Prevents tracing of 'administrative events such as<br>• PL/SQL Trace Tool started<br>• Trace flags changed<br>• PL/SQL Virtual Machine started<br>• PL/SQL Virtual Machine stopped |
| NO_TRACE_HANDLED_EXCEPTIONS | INTEGER | 65536 | Prevents tracing of handled exceptions |

**Table 204-2 DBMS_TRACE Version Constants**

| Name | Type | Value | Description |
|---|---|---|---|
| TRACE_MINOR_VERSION | INTEGER | 0 | |
| TRACE_MAJOR_VERSION | INTEGER | 1 | |

Oracle recommends using the symbolic form for all these constants.

# DBMS_TRACE Restrictions

You cannot use PL/SQL tracing in a shared server environment.

# DBMS_TRACE Operational Notes

Certain operational notes apply to `DBMS_TRACE`.

These are described in the following sections:



### Controlling Data Volume

Profiling large applications may produce a large volume of data. You can control the volume of data collected by *enabling* specific program units for trace data collection.

You can enable a program unit by compiling it debug. This can be done in one of two ways:

```
alter session set plsql_debug=true;
create or replace ... /* create the library units - debug information will be generated
*/
```

or:

```
/* recompile specific library unit with debug option */
alter [PROCEDURE | FUNCTION | PACKAGE BODY]  <libunit-name> compile debug;
```

> **Note:**
>
> You cannot use the second method for anonymous blocks.

You can limit the amount of storage used in the database by retaining only the most recent 8,192 records (approximately) by including `TRACE_LIMIT` in the `TRACE_LEVEL` parameter of the `SET_PLSQL_TRACE` procedure.

### Creating Database Tables to Collect DBMS_TRACE Output

You must create database tables into which the `DBMS_TRACE` package writes output. Otherwise, the data is not collected. To create these tables, run the script `TRACETAB.SQL`. The tables this script creates are owned by SYS.

### Collecting Trace Data

The PL/SQL features you can trace are described in the script `DBMSPBT.SQL`. Some of the key tracing features are:

- Tracing Calls
- Tracing Exceptions
- Tracing SQL
- Tracing Lines

Additional features of DBMS_TRACE also allow pausing and resuming trace, and limiting the output.

### Tracing Calls

Two levels of call tracing are available:

- Level 1: Trace all calls. This corresponds to the constant `TRACE_ALL_CALLS`.
- Level 2: Trace calls to enabled program units only. This corresponds to the constant `TRACE_ENABLED_CALLS`.

Enabling cannot be detected for remote procedure calls (RPCs); hence, RPCs are only traced with level 1.

### Tracing Exceptions

Two levels of exception tracing are available:

- Level 1: Trace all exceptions. This corresponds to `TRACE_ALL_EXCEPTIONS`.
- Level 2: Trace exceptions raised in enabled program units only. This corresponds to `TRACE_ENABLED_EXCEPTIONS`.

### Tracing SQL

Two levels of SQL tracing are available:

- Level 1: Trace all SQL. This corresponds to the constant `TRACE_ALL_SQL`.
- Level 2: Trace SQL in enabled program units only. This corresponds to the constant `TRACE_ENABLED_SQL`.

### Tracing Lines

Two levels of line tracing are available:

- Level 1: Trace all lines. This corresponds to the constant `TRACE_ALL_LINES`.
- Level 2: Trace lines in enabled program units only. This corresponds to the constant `TRACE_ENABLED_LINES`.

When tracing lines, Oracle adds a record to the database each time the line number changes. This includes line number changes due to procedure calls and returns.

> **Note:**
>
> For all types of tracing, level 1 overrides level 2. For example, if both level 1 and level 2 are enabled, then level 1 takes precedence.

### Collected Data

If tracing is requested only for enabled program units, and if the current program unit is not enabled, then no trace data is written.

When tracing calls, both the call and return are traced. The check for whether tracing is "enabled" passes if either the called routine or the calling routine is "enabled".

Call tracing will always output the program unit type, program unit name, and line number for both the caller and the callee. It will output the caller's stack depth. If the caller is enabled, the caller's name will also be output. If the callee is enabled, the callee's name will also be output.

Exception tracing writes out the line number. Raising the exception shows information on whether the exception is user-defined or pre-defined. It also shows the exception number in the case of pre-defined exceptions. Both the place where the exceptions are raised and their handler is traced. The check for tracing being "enabled" is done independently for the place where the exception is raised and the place where the exception is handled. Enabling `NO_TRACE_HANDLED_EXCEPTIONS` limits data collection to unhandled exceptions

All calls to `DBMS_TRACE.SET_PLSQL_TRACE` and `DBMS_TRACE.CLEAR_PLSQL_TRACE` place a special trace record in the database. Therefore, it is always possible to determine when trace settings were changed.

#### Trace Control

As well as determining which items are collected, you can pause and resume the trace process. No information is gathered between the time that tracing is paused and the time that it is resumed. The constants `TRACE_PAUSE` and `TRACE_RESUME` are used to accomplish this. Trace records are generated to indicate that the trace was paused/resumed.

It is also possible to retain only the last 8,192 trace events of a run by using the constant `TRACE_LIMIT`. This allows tracing to be turned on without filling up the database. When tracing stops, the last 8,192 records are saved. The limit is approximate, since it is not checked on every trace record. At least the requested number of trace records will be generated; up to 1,000 additional records may be generated. At least the requested number of trace records will be generated; up to 1,000 additional records may be generated. The 8,192 record limit can be changed. Setting event 10940 to level *n* changes the record limit to 1024 * *n*.

Enabling `NO_TRACE_ADMINISTRATIVE` prevents the generation of such administrative event records as `PL/SQL Trace Tool started`, `Trace flags changed`, `PL/SQL Virtual Machine started`, and `PL/SQL Virtual Machine stopped`.

# Summary of DBMS_TRACE Subprograms

This table lists the `DBMS_TRACE` subprograms and briefly describes them.

**Table 204-3 DBMS_TRACE Package Subprograms**

| Subprogram | Description |
|---|---|
| CLEAR_PLSQL_TRACE Procedure | Stops trace data dumping in session |
| GET_PLSQL_TRACE_LEVEL Function | Gets the trace level |
| PLSQL_TRACE_VERSION Procedure | Gets the version number of the trace package |
| SET_PLSQL_TRACE Procedure | Starts tracing in the current session |

# CLEAR_PLSQL_TRACE Procedure

This procedure disables trace data collection.

#### Syntax

```
DBMS_TRACE.CLEAR_PLSQL_TRACE;
```

# GET_PLSQL_TRACE_LEVEL Function

This procedure returns the current trace level as the sum of one or more DBMS_TRACE constants.

See Table 204-1 for a list of the constants.

### Syntax

```
DBMS_TRACE.GET_PLSQL_TRACE_LEVEL
  RETURN BINARY_INTEGER;
```

# PLSQL_TRACE_VERSION Procedure

This procedure gets the version number of the trace package. It returns the major and minor version number of the DBMS_TRACE package.

### Syntax

```
DBMS_TRACE.PLSQL_TRACE_VERSION (
   major OUT BINARY_INTEGER,
   minor OUT BINARY_INTEGER);
```

### Parameters

**Table 204-4 PLSQL_TRACE_VERSION Procedure Parameters**

| Parameter | Description |
|---|---|
| major | Major version number of DBMS_TRACE. |
| minor | Minor version number of DBMS_TRACE. |

# SET_PLSQL_TRACE Procedure

This procedure enables PL/SQL trace data collection.

### Syntax

```
DBMS_TRACE.SET_PLSQL_TRACE (
   trace_level INTEGER);
```

### Parameters

**Table 204-5 *SET_PLSQL_TRACE Procedure Parameters***

| Parameter | Description |
|---|---|
| trace_level | You must supply one or more of the constants as listed in Table 204-1. By summing the constants, you can enable tracing of multiple PL/SQL language features simultaneously. The control constants "TRACE_PAUSE", "TRACE_RESUME" and "TRACE_STOP" should not be used in combination with other constants.<br>Also see DBMS_TRACE Operational Notes: Collecting Trace Data for more information. |