61
DBMS_CUBE_LOG

DBMS_CUBE_LOG contains subprograms for creating and managing logs for cubes and cube dimensions.

> **See Also:**
>
> OLAP Technology in the Oracle Database in*Oracle OLAP User's Guide* regarding use of the OLAP option to support business intelligence and analytical applications.

This chapter contains the following topics:

- Using DBMS_CUBE_LOG
- Summary of DBMS_CUBE_LOG Subprograms

Using DBMS_CUBE_LOG

DBMS_CUBE_LOG manages several logs that enable you to track the progress of long running processes, then use the results to profile performance characteristics.

They provide information to help you diagnose and remedy problems that may occur during development and maintenance of a cube, such as hierarchies that are improperly structured in the relational source tables, records that fail to load, or data refreshes that take too long to complete. They also help diagnose performance problems in querying cubes.

Analytic Workspace Manager creates the logs automatically using the default names and types. It also disables the logs when Analytic Workspace Manager is closed. To use the same logs outside of Analytic Workspace Manager, you must first enable them. Alternatively, you can create and manage different logs for use outside of Analytic Workspace Manager.

This section contains the following topics:

- Logging Types
- Logging Targets
- Verbosity Levels
- Security Model
- Creating Cube Logs
- Cube Build Log
- Cube Dimension Compile Log
- Cube Operations Log
- Cube Rejected Records Log

DBMS_CUBE_LOG — Logging Types

Several logs are available, each one dedicated to storing messages of a particular type. You may use all of them or only those that you find particularly valuable.

The logs and their contents are described later in this topic.

- Cube Build Log
- Cube Dimension Compile Log
- Cube Operations Log
- Cube Rejected Records Log

`DBMS_CUBE_LOG` provides functions that return the binary integer for each log type. You can produce more readable code by using these functions instead of integers for the argument values of other `DBMS_CUBE_LOG` procedures and functions. Refer to these descriptions:

- TYPE_BUILD Function
- TYPE_DIMENSION_COMPILE Function
- TYPE_OPERATIONS Function
- TYPE_REJECTED_RECORDS Function

DBMS_CUBE_LOG — Logging Targets

The `TABLE_CREATE` procedure creates database tables for storing the logs. Using the `ENABLE` procedure, you can create additional targets with changes in the destination or logging level. For example, you might target the Cube Operations log to both a table and a disk file.

These are the available targets:

- Disk file
- LOB
- Database table
- Trace file

See ENABLE Procedure for more information about creating multiple targets.

`DBMS_CUBE_LOG` provides functions that return the binary integer for each target type. You can produce more readable code by using these functions instead of integers for the argument values of other `DBMS_CUBE_LOG` procedures and functions. Refer to these descriptions:

- TARGET_FILE Function
- TARGET_LOB Function
- TARGET_TABLE Function
- TARGET_TRACE Function

DBMS_CUBE_LOG — Verbosity Levels

You can decide how much information is recorded in a log. You may want fewer details when leaving a job to run overnight than when you are monitoring the success of a new build. You can choose from these verbosity levels. Each level adds to the preceding level.

- `LOWEST`: Logs the status of each command used to build the cube dimensions and cubes, the use of slave processes, and summary records. This is the basic logging level.
- `LOW`: Logs messages from the OLAP engine, such as start and finish records for SQL Import, Aggregate, and Update.
- `MEDIUM`: Logs messages at the level used by Analytic Workspace Manager.
- `HIGH`: Logs messages that provide tuning information, such as composite lengths, partitioning details, object sizes, and aggregation work lists. This level is intended for use by Oracle Field Services.
- `HIGHEST`: Logs debugging messages and other information typically sent to a trace file. This level is intended for use by Oracle Support Services.

`DBMS_CUBE_LOG` provides functions that return the binary integer for each verbosity level. You can produce more readable code by using these functions instead of integers for the argument values of other `DBMS_CUBE_LOG` procedures and functions. Refer to these descriptions:

- LEVEL_LOWEST Function
- LEVEL_LOW Function
- LEVEL_MEDIUM Function
- LEVEL_HIGH Function
- LEVEL_HIGHEST Function

DBMS_CUBE_LOG Security Model

The `TABLE_CREATE` procedure requires the `CREATE TABLE` privilege.

DBMS_CUBE_LOG — Creating Cube Logs

To store logging information in a database table, you must create that table using the TABLE_CREATE procedure. Cube Build logs are always stored in tables. The ENABLE procedure creates the other target types for the other logs.

To create a Cube Build log:

- Execute the `TABLE_CREATE` procedure.

The following command creates a Cube Build log with the default name of `CUBE_BUILD_LOG`:

```
EXECUTE dbms_cube_log.table_create(dbms_cube_log.type_build);
```

To create a Cube Dimension Compile log, Cube Operations log, or Cube Rejected Records log with a database table target:

1. Execute the `TABLE_CREATE` procedure to create the table.
2. Execute the `ENABLE` procedure to begin logging.

These commands create and enable a Cube Operations table with the default name of CUBE_OPERATIONS_LOG and the default verbosity level:

```
EXECUTE dbms_cube_log.table_create(dbms_cube_log.type_operations);
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_operations);
```

To create a Cube Dimension Compile log, Cube Operations log, or Cube Rejected Records log with a trace file, disk file, or LOB target:

- Execute the ENABLE procedure.

This command enables the Cube Rejected Records log, sets verbosity to the lowest level, and directs the output to a disk file named rejects.log in the WORK_DIR database directory:

```
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_rejected_records, -
     dbms_cube_log.target_file, dbms_cube_log.level_lowest, -
     'WORK_DIR/rejects.log');
```

DBMS_CUBE_LOG — Cube Build Log

The Cube Build log provides information about what happened during a build. Use this log to determine whether the build produced the results you were expecting, and if not, why not.

The contents of the Cube Build log are refreshed continuously during a build. You can query the log at any time to evaluate the progress of the build and to estimate the time to completion.

The default name of the Cube Build log is CUBE_BUILD_LOG. The following table describes its contents.

> **Note:**
>
> To manage a Cube Build log, use only the TABLE_CREATE and VERSION procedures.

Column	Datatype	NULL	Description
BUILD_ID	NUMBER	--	A unique sequence number for the build. The same number is used for slave processes in a parallel build.
SLAVE_NUMBER	NUMBER	--	A counter for slave processes in a parallel build: 0 is the master process, 1 is the first slave, 2 is the second slave, and so forth.
STATUS	VARCHAR2(10)	--	The current status of the command: STARTED, COMPLETED, FAILED, or WORKING.
COMMAND	VARCHAR2(25)	--	The name of the command being executed, such as BUILD, LOAD, and SOLVE.
BUILD_OBJECT	VARCHAR2(500)	--	The name of the cube or cube dimension being processed.
BUILD_OBJECT_TYPE	VARCHAR2(10)	--	The type of object: CUBE, DIMENSION, or BUILD.
OUTPUT	CLOB	--	Information structured like an XML document about the command, or NULL when there is no additional information, such as for a STARTED row.

Column	Datatype	NULL	Description
AW	VARCHAR2(30)	--	The name of the analytic workspace that contains the objects of the build.
OWNER	VARCHAR2(30)		The owner of the analytic workspace and all the objects of the build.
PARTITION	VARCHAR2(10)	--	The name of the partition being processed, or NULL when the current operation does not correspond to a partition.
SCHEDULER_JOB	VARCHAR2(100)	--	A user-specified string to identify the build.
TIME	TIMESTAMP(6)	--	The time the row is added to the table.
BUILD_SCRIPT	CLOB	--	The cube build script. Populated only in rows where COMMAND is BUILD.
BUILD_TYPE	VARCHAR2(22)	--	The origin of the build: DBMS_CUBE, DBMS_MVIEW, JAVA, or SLAVE.
COMMAND_DEPTH	NUMBER(2)	--	The nesting depth of the command. For example, COMPILE HIERARCHIES is a component step of COMPILE, so if COMPILE has a depth of 1, then COMPILE HIERARCHIES has a depth of 2.
BUILD_SUB_OBJECT	VARCHAR2(30)	--	The name of a subobject being processed, such as a measure that does not inherit the aggregation rules of the cube.
REFRESH_METHOD	VARCHAR2(1)	--	The refresh method, such as C or F, that is associated with the current command. The refresh method is important only for the CLEAR step.
SEQ_NUMBER	NUMBER	--	Not currently used.
COMMAND_NUMBER	NUMBER	--	The sequence number of the command in the current process, which can be used to distinguish the same command on different objects. For example, a LOAD on PRODUCT and a LOAD on TIME.
IN_BRANCH	NUMBER(1)	--	Not currently used.
COMMAND_STATUS_NUMBER	NUMBER	--	Identifies the sequence number of all rows for a particular command. For example, a particular command might be represented by four rows: The first row has a status of STARTED and the last row has a status of COMPLETED. This column is used for sorting.

DBMS_CUBE_LOG — Cube Dimension Compile Log

When solving a cube, OLAP checks the dimension hierarchies to make sure they are valid. Errors that occur during this validation are written to the Cube Dimension Compile log.

The checks include:

- Circularity: Hierarchies are defined by parent-child relations among dimension members. Circularity occurs when a dimension member is specified as its own ancestor or descendant.

- Hierarchy type: Hierarchies can be level based or value based. You can define a cube so that only level-based hierarchies are valid, such as a cube materialized view.
- Level options: Level-based hierarchies can be regular, ragged, or skip level. You can define a dimension so that only regular hierarchies are valid, such as a Time dimension.

The default name of the Cube Dimension Compile log is `CUBE_DIMENSION_COMPILE`. The following table describes its contents.

Column	Datatype	NULL	Description
`ID`	`NUMBER`	--	Current operation identifier
`SEQ_NUMBER`	`NUMBER`	--	Sequence number in the Cube Build log
`ERROR#`	`NUMBER(8)`	`NOT NULL`	Number of the error being reported
`ERROR_MESSAGE`	`VARCHAR2(2000)`	--	Error message associated with the error
`DIMENSION`	`VARCHAR2(100)`	--	Name of the dimension being compiled
`DIMENSION_MEMBER`	`VARCHAR2(100)`	--	Faulty dimension member
`MEMBER_ANCESTOR`	`VARCHAR2(100)`	--	Parent of `DIMENSION_MEMBER`
`HIERARCHY1`	`VARCHAR2(100)`	--	First hierarchy involved in the error
`HIERARCHY2`	`VARCHAR2(100)`	--	Second hierarchy involved in the error
`ERROR_CONTEXT`	`CLOB`	--	Additional information about the error

DBMS_CUBE_LOG — Cube Operations Log

The Cube Operations log contains messages and debugging information for all OLAP engine events. You can track current operations at a very detailed level. Using the `SQL_ID` column, you can join the Cube Operations log to dynamic performance views such as `V$SQL`, `V$SESSION`, and `V$SESSION_LONGOPS` to see cube operations in the context of other database operations such as I/O Wait and CPU.

The default name of the Cube Operations log is `CUBE_OPERATIONS_LOG`. The following table describes its contents.

Column	Datatype	NULL	Description
`INST_ID`	`NUMBER`	`NOT NULL`	Instance identifier
`SID`	`NUMBER`	`NOT NULL`	Session identifier
`SERIAL#`	`NUMBER`	`NOT NULL`	Session serial number
`USER#`	`NUMBER`	`NOT NULL`	User identifier
`SQL_ID`	`VARCHAR2(13)`	--	Executing SQL statement identifier
`JOB`	`NUMBER`	--	Job identifier
`ID`	`NUMBER`	--	Current operation identifier
`PARENT_ID`	`NUMBER`	--	Parent operation identifier
`SEQ_NUMBER`	`NUMBER`	--	Sequence number in the Cube Build log
`TIME`	`TIMESTAMP(6) WITH TIME ZONE`	`NOT NULL`	Time the record was added to the Cube Operations log
`LOG_LEVEL`	`NUMBER(4)`	`NOT NULL`	Verbosity level of the record, as specified by the `DBMS_CUBE_LOG.ENABLE` procedure.

Column	Datatype	NULL	Description
DEPTH	NUMBER(4)	--	Nesting depth of the record. For example, a level of 0 indicates that the operation and suboperation are not nested within other operations and suboperations.
OPERATION	VARCHAR2(15)	NOT NULL	Current operation, such as AGGREGATE, ROWSOURCE, or SQLIMPORT.
SUBOPERATION	VARCHAR2(20)	--	Current suboperation, such as Loading or Import
STATUS	VARCHAR2(10)	NOT NULL	Current status of the operation, such as START, TRACE, COMPLETED, or Failed.
NAME	VARCHAR2(20)	NOT NULL	Name of the record, such as ROWS LOADED, AVE_ROW_LEN, and PAGEPOOLSIZE
VALUE	VARCHAR2(4000)	--	Value of NAME
DETAILS	CLOB	--	Additional information about NAME.

DBMS_CUBE_LOG — Cube Rejected Records Log

The Cube Rejected Records log contains a summary of the loader job and any records that were rejected because they did not meet the expected format.

A single row in the source table may have errors in more than one field. Each field generates an error in the log, resulting in multiple rows with the same rowid4

in the SOURCE_ROW column.

The default name of the Cube Rejected Records log is CUBE_REJECTED_RECORDS. The following table describes its contents.

Column	Datatype	NULL	Description
ID	NUMBER	--	Current operation identifier
SEQ_NUMBER	NUMBER	--	Sequence number in the Cube Build log
ERROR#	NUMBER(8)	NOT NULL	Number of the error triggered by the record
ERROR_MESSAGE	VARCHAR2	--	Error message associated with the error
RECORD#	NUMBER(38)	--	Input record number
SOURCE_ROW	ROWID	--	Rowid of the row in the source table; null when the source is a view or a query

Summary of DBMS_CUBE_LOG Subprograms

This table lists and describes the DBMS_CUBE_LOG subprograms.

Table 61-1 DBMS_CUBE_LOG Subprograms

Subprogram	Description
DEFAULT_NAME Function	Returns the default table names of the various log types.
DISABLE Procedure	Turns logging off for the duration of a session.
ENABLE Procedure	Turns on logging for the duration of a session, redirects logging to additional output types, and changes the verbosity level in the logs.

Table 61-1 (Cont.) DBMS_CUBE_LOG Subprograms

Subprogram	Description
FLUSH Procedure	Forces all buffered messages to be written to the logs.
GET_LOG Procedure	Returns the current settings for the level and location of a particular log.
GET_LOG_SPEC Function	Retrieves a description of all active logs.
GET_PARAMETER Function	Returns the current values of the options that control various aspects of logging.
LEVEL_HIGH Function	Returns the integer value of the high verbosity level.
LEVEL_HIGHEST Function	Returns the integer value of the highest verbosity level.
LEVEL_LOW Function	Returns the integer value of the low verbosity level.
LEVEL_LOWEST Function	Returns the integer value of the lowest verbosity level.
LEVEL_MEDIUM Function	Returns the integer value of the medium verbosity level.
SET_LOG_SPEC Procedure	Sets all logging to the values specified in the input string.
SET_PARAMETER Procedure	Sets options that control various aspects of logging.
TABLE_CREATE Procedure	Creates the table targets for the OLAP logs.
TARGET_FILE Function	Returns the integer value of a disk file target.
TARGET_LOB Function	Returns the integer value of a LOB target.
TARGET_TABLE Function	Returns the integer value of a database table target
TARGET_TRACE Function	Returns the integer value of a trace file target.
TYPE_BUILD Function	Returns the integer value of the Cube Build log.
TYPE_DIMENSION_COMPILE Function	Returns the integer value of the Cube Dimension Compile log.
TYPE_OPERATIONS Function	Returns the integer value of the Cube Operations log.
TYPE_REJECTED_RECORDS Function	Returns the integer value of the Cube Rejected Records log.
VERSION Function	Returns the version number of a specific log table or the current version number of a specific log type.

DEFAULT_NAME Function

This function returns the default table names of the various log types.

Syntax

```
DBMS_CUBE_LOG.DEFAULT_NAME (
          LOG_TYPE       IN   BINARY_INTEGER  DEFAULT TYPE_OPERATIONS)
    RETURN VARCHAR2;
```

Parameters

Table 61-2 DEFAULT_NAME Function Parameters

Parameter	Description
log_type	One of the following log types: • 1: TYPE_OPERATIONS • 2: TYPE_REJECTED_RECORDS • 3: TYPE_DIMENSION_COMPILE • 4: TYPE_BUILD See "Logging Types".

Returns

The default table name of the specified log type.

Examples

This example returns the default name of the Cube Operations log:

```
SELECT dbms_cube_log.default_name FROM dual;

DEFAULT_NAME
------------------------------
CUBE_OPERATIONS_LOG
```

The next example returns the default name of the Cube Rejected Records log:

```
select dbms_cube_log.default_name(dbms_cube_log.type_rejected_records) -
     "Default Name" from dual;

Default Name
-------------------------
CUBE_REJECTED_RECORDS
```

DISABLE Procedure

This procedure turns logging off for the duration of a session, unless logging is explicitly turned on again with the ENABLE procedure.

Syntax

```
DBMS_CUBE_LOG.DISABLE (
          LOG_TYPE       IN   BINARY_INTEGER  DEFAULT,
          LOG_TARGET     IN   BINARY_INTEGER  DEFAULT);
```

Parameters

Table 61-3 DISABLE Procedure Parameters

Parameter	Description
log_type	One of the following log types: • 1: TYPE_OPERATIONS • 2: TYPE_REJECTED_RECORDS • 3: TYPE_DIMENSION_COMPILE **Note**: You cannot disable the Cube Build log with this procedure. See "Logging Types".
log_target	One of the following destinations for the logging records. The logs are sent to a table unless you previously specified a different target using the ENABLE procedure. • 1: TARGET_TABLE • 2: TARGET_TRACE • 3: TARGET_FILE • 4: TARGET_LOB See "Logging Targets"

Example

This command disables the dimension compilation error log table:

```
EXECUTE dbms_cube_log.disable(dbms_cube_log.type_dimension_compile);
```

ENABLE Procedure

This procedure turns on logging for the duration of a session or until it is turned off using the DISABLE procedure.

The ENABLE procedure also allows you to direct logging to additional output types and to change the amount of detail in the logs. You can enable a log type to each of the log targets. For example, you can enable the Cube Operations log to the trace file, a table, and a file at different verbosity levels, but you cannot enable the Cube Operations log to two files at the same time.

This procedure also checks the format of the logs and updates them if necessary.

Syntax

```
DBMS_CUBE_LOG.ENABLE (
          LOG_TYPE       IN     BINARY_INTEGER  DEFAULT NULL,
          LOG_TARGET     IN     BINARY_INTEGER  DEFAULT NULL,
          LOG_LEVEL      IN     BINARY_INTEGER  DEFAULT NULL);

DBMS_CUBE_LOG.ENABLE (
          LOG_TYPE       IN     BINARY_INTEGER  DEFAULT NULL,
          LOG_TARGET     IN     BINARY_INTEGER  DEFAULT NULL,
          LOG_LEVEL      IN     BINARY_INTEGER  DEFAULT NULL,
          LOG_LOCATION   IN     VARCHAR2        DEFAULT NULL);

DBMS_CUBE_LOG.ENABLE (
          LOG_TYPE       IN     BINARY_INTEGER  DEFAULT NULL,
          LOG_TARGET     IN     BINARY_INTEGER  DEFAULT NULL,
```

```
LOG_LEVEL      IN     BINARY_INTEGER  DEFAULT NULL,
LOG_LOCATION   IN/OUT CLOB );
```

Parameters

Table 61-4 ENABLE Procedure Parameters

Parameter	Description
log_type	One of the following log types: • 1: TYPE_OPERATIONS • 2: TYPE_REJECTED_RECORDS • 3: TYPE_DIMENSION_COMPILE **Note**: You cannot disable the Cube Build log with this procedure. See "Logging Types".
log_target	One of the following destinations for the logging records. The logs are sent to a table unless you previously specified a different target using the ENABLE procedure. • 1: TARGET_TABLE • 2: TARGET_TRACE • 3: TARGET_FILE • 4: TARGET_LOB See "Logging Targets"
log_level	One of the following log verbosity levels. Each level adds new types of messages to the previous level. • 1: LEVEL_LOWEST • 2: LEVEL_LOW • 3: LEVEL_MEDIUM • 4: LEVEL_HIGH • 5: LEVEL_HIGHEST See "Verbosity Levels".
log_location	The full identity of the log, such as *owner.table_name* when log_target is a table.

Examples

The following command enables all cube logs:

```
EXECUTE dbms_cube_log.enable;
```

The following PL/SQL procedure sets the log level to LEVEL_LOWEST:

```
BEGIN
     dbms_cube_log.disable(dbms_cube_log.type_rejected_records);
     dbms_cube_log.enable(dbms_cube_log.type_rejected_records,
          dbms_cube_log.target_table, dbms_cube_log.level_lowest);
END;
/
```

FLUSH Procedure

This procedure forces all buffered messages to be written to the logs.

The buffers are flushed automatically throughout a session, but manually flushing them before viewing the logs assures that you can view all of the messages.

Syntax

```
DBMS_CUBE_LOG.FLUSH ( );
```

Example

The following example flushes the buffers for all of the logs:

```
EXECUTE dbms_cube_log.flush;
```

GET_LOG Procedure

This procedure returns the current settings for the level and location of a particular log.

Syntax

```
DBMS_CUBE_LOG.GET_LOG (
          LOG_TYPE       IN   BINARY_INTEGER  DEFAULT TYPE_OPERATIONS,
          LOG_TARGET     IN   BINARY_INTEGER  DEFAULT TARGET_TABLE,
          LOG_LEVEL      OUT  BINARY_INTEGER,
          LOG_LOCATION   OUT  VARCHAR2 );
```

Parameters

Table 61-5 GET_LOG Procedure Parameters

Parameter	Description
log_type	One of the following log types: • 1: TYPE_OPERATIONS • 2: TYPE_REJECTED_RECORDS • 3: TYPE_DIMENSION_COMPILE See "Logging Types".
log_target	One of the following destinations for the logging records. The logs are sent to a table unless you previously specified a different target using the ENABLE procedure. • 1: TARGET_TABLE • 2: TARGET_TRACE • 3: TARGET_FILE • 4: TARGET_LOB See "Logging Targets"
log_level	One of the following log verbosity levels. Each level adds new types of messages to the previous level. • 1: LEVEL_LOWEST • 2: LEVEL_LOW • 3: LEVEL_MEDIUM • 4: LEVEL_HIGH • 5: LEVEL_HIGHEST See "Verbosity Levels".
log_location	The full identity of the log, such as *owner.table_name* when log_target is a table.

Usage Notes

If `log_type` is not active, then `log_level` and `log_location` are null. Use `DBMS_CUBE_LOG.ENABLE` to activate a log.

Examples

This PL/SQL procedure provides information about the Cube Rejected Records log:

```
SET serverout ON format wrapped

DECLARE
     myloglevel  binary_integer;
     mylogtarget varchar2(128);

BEGIN
     dbms_cube_log.get_log(dbms_cube_log.type_rejected_records,
     dbms_cube_log.target_table, myloglevel, mylogtarget);

     dbms_output.put_line('Log Level: ' || myloglevel);
     dbms_output.put_line('Log Target: ' || mylogtarget);
END;
/
```

The procedure generates results like the following:

```
Log Level: 5

Log Target: GLOBAL.CUBE_REJECTED_RECORDS
```

GET_LOG_SPEC Function

This function retrieves a description of all active Cube Operations logs, Cube Rejected Records logs, and Cube Dimension Compile logs.

Syntax

```
DBMS_CUBE_LOG.GET_LOG_SPEC ( )
     RETURN VARCHAR2;
```

Returns

The type and target of all active logs.

Usage Notes

You can use the output from this function as the input to `SET_LOG_SPEC`.

Examples

The following example shows that the Cube Operations log, Cube Rejected Records log, and Cube Dimension Compile log are active. The Cube Operations log is stored in the session trace file and the other logs are stored in tables.

```
SELECT dbms_cube_log.get_log_spec FROM dual;

GET_LOG_SPEC
--------------------------------------------------------------------------------
OPERATIONS(TABLE, TRACE) REJECTED_RECORDS(TABLE[DEBUG])
```

GET_PARAMETER Function

This function returns the current values of the options that control various aspects of logging. To set these options, use the `SET_PARAMETER` function.

Syntax

```
DBMS_CUBE_LOG.GET_PARAMETER (
          LOG_TYPE       IN   BINARY_INTEGER,
          LOG_PARAMETER  IN   BINARY_INTEGER )
     RETURN BINARY_INTEGER;
```

Parameters

Table 61-6 GET_PARAMETER Function Parameters

Parameter	Description
`log_type`	One of the following log types: • 1: `TYPE_OPERATIONS` • 2: `TYPE_REJECTED_RECORDS` • 3: `TYPE_DIMENSION_COMPILE` See "Logging Types".
`log_parameter`	One of the following options: • 1: `MAX_ERRORS` • 2: `FLUSH_INTERVAL` • 3: `LOG_FULL_RECORD` • 4: `LOG_EVERY_N` • 5: `ALLOW_ERRORS` See "SET_PARAMETER Procedure".

Returns

The value of the specified *`log_parameter`*.

Examples

This example shows the current maximum number of errors in the Cube Rejected Records log before logging stops. This parameter was previously set with the `SET_PARAMETER` procedure.

```
SELECT dbms_cube_log.get_parameter(dbms_cube_log.type_rejected_records, 1) -
     "Maximum Records" FROM dual;

Maximum Records
---------------
            100
```

LEVEL_HIGH Function

This function returns the integer value of the high verbosity level.

Syntax

```
DBMS_CUBE_LOG.LEVEL_HIGH ()
     RETURN BINARY_INTEGER;
```

Returns

4

Usage Notes

Use this function instead of its binary integer equivalent for the *LOG_LEVEL* parameter in DBMS_CUBE_LOG subprograms. See "Verbosity Levels".

Example

This command sets the verbosity level of the cube operations table to high:

```
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_operations, -
     dbms_cube_log.target_table, dbms_cube_log.level_high);
```

LEVEL_HIGHEST Function

This function returns the integer value of the highest verbosity level.

Syntax

```
DBMS_CUBE_LOG.LEVEL_HIGHEST ()
     RETURN BINARY_INTEGER;
```

Returns

5

Usage Notes

Use this function instead of its binary integer equivalent for the LOG_LEVEL parameter in DBMS_CUBE_LOG subprograms. See "Verbosity Levels".

Example

This command sets the verbosity level of the cube operations table to highest:

```
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_operations, -
     dbms_cube_log.target_table, dbms_cube_log.level_highest);
```

LEVEL_LOW Function

This function returns the integer value of the low verbosity level.

Syntax

```
DBMS_CUBE_LOG.LEVEL_LOW ()
     RETURN BINARY_INTEGER;
```

Returns

2

Usage Notes

Use this function instead of its binary integer equivalent for the LOG_LEVEL parameter in DBMS_CUBE_LOG subprograms. See "Verbosity Levels".

Example

This command sets the verbosity level of the cube operations table to low:

```
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_operations, -
     dbms_cube_log.target_table, dbms_cube_log.level_low);
```

LEVEL_LOWEST Function

This function returns the integer value of the lowest verbosity level. This level logs the status of each command used to build the cube dimensions and cubes, the use of slave processes, and summary records.

Syntax

```
DBMS_CUBE_LOG.LEVEL_LOWEST ()
     RETURN BINARY_INTEGER;
```

Returns

```
1
```

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_LEVEL` parameter in `DBMS_CUBE_LOG` subprograms. See "Verbosity Levels".

Example

This command sets the verbosity level of the cube operations table to lowest:

```
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_operations, -
     dbms_cube_log.target_table, dbms_cube_log.level_lowest);
```

LEVEL_MEDIUM Function

This function returns the integer value of the medium verbosity level.

Syntax

```
DBMS_CUBE_LOG.LEVEL_MEDIUM ()
     RETURN BINARY_INTEGER;
```

Returns

```
3
```

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_LEVEL` parameter in `DBMS_CUBE_LOG` subprograms. See "Verbosity Levels".

Example

This command sets the verbosity level of the cube operations table to medium:

```
EXECUTE dbms_cube_log.enable(dbms_cube_log.type_operations, -
     dbms_cube_log.target_table, dbms_cube_log.level_medium);
```

SET_LOG_SPEC Procedure

This procedure sets all logging to the values specified in the input string.

Syntax

```
DBMS_CUBE_LOG.SET_LOG_SPEC (
          LOG_SPEC       IN   VARCHAR2 );
```

Parameters

Table 61-7 SET_LOG_SPEC Procedure Parameters

Parameter	Description
log_spec	A string consisting of *type*(*target*) pairs. *Type* can be: • OPERATIONS • REJECTED_RECORDS • DIMENSION_COMPILE *Target* can be: • TABLE • TRACE • FILE • LOB

Usage Notes

The GET_LOG_SPEC function returns a properly formatted string for SET_LOG_SPEC.

Examples

This PL/SQL procedure disables all logs, verifies that they are disabled, then activates the Cube Operations log and the Cube Rejected Records log.

```
BEGIN
     dbms_cube_log.disable;
     dbms_output.put_line('Cube Logs: ' || dbms_cube_log.get_log_spec);

     dbms_cube_log.set_log_spec('OPERATIONS(TRACE) REJECTED_RECORDS(TABLE)');
     dbms_output.put_line('Cube Logs: ' || dbms_cube_log.get_log_spec);
END;
/
```

The output from the procedure verifies that the DISABLE function de-activated all logs, and the SET_LOG_SPEC function activated two logs:

```
Cube Logs:

Cube Logs: OPERATIONS(TRACE) REJECTED_RECORDS(TABLE)
```

SET_PARAMETER Procedure

This procedure sets options that control various aspects of logging.

To obtain the current value of these options, use the GET_PARAMETER function.

Syntax

```
DBMS_CUBE_LOG.SET_PARAMETER (
          LOG_TYPE       IN   BINARY_INTEGER,
          LOG_PARAMETER  IN   BINARY_INTEGER,
          VALUE          IN   BINARY_INTEGER );
```

Parameters

Table 61-8 SET_PARAMETER Procedure Parameters

Parameter	Description
log_type	One of the following log types: • 1: TYPE_OPERATIONS • 2: TYPE_REJECTED_RECORDS • 3: TYPE_DIMENSION_COMPILE • 4: TYPE_BUILD See "Logging Types".
log_parameter	One of the following parameters: • 1: MAX_ERRORS Maximum number of records before signalling an end to logging, such as the number of rejected records in the Cube Rejected Records log or the number of compilation errors in the dimension compilation error log. • 2: FLUSH_INTERVAL The number of seconds to buffer the records before writing them to a log. When this parameter is 0, the records are written directly to the logs without buffering. • 3: LOG_FULL_RECORD Controls logging of rejected records. Set this parameter to one of the following constants: 0: FULL_RECORD_AUTO: Log the full record when no row ID is available. 1: FULL_RECORD_ALWAYS: Always log the full record. 2: FULL_RECORD_NEVER: Never log the full record. • 4: LOG_EVERY_N Enters a progress message every *n* rows during data maintenance. • 5: ALLOW_ERRORS: Displays logging errors, which are initially turned off to allow processing to proceed.
value	The new value of *log_parameter*.

Examples

This PL/SQL procedure sets the two parameters, then uses the GET_PARAMETER function to show the settings:

```
BEGIN
     dbms_cube_log.set_parameter(dbms_cube_log.type_rejected_records, 1, 150);
     dbms_cube_log.set_parameter(dbms_cube_log.type_rejected_records, 2, 5);

     dbms_output.put_line('Max rejected records: ' ||
        dbms_cube_log.get_parameter(dbms_cube_log.type_rejected_records, 1));
```

```
     dbms_output.put_line('Buffer time: ' ||
        dbms_cube_log.get_parameter(dbms_cube_log.type_rejected_records, 2) ||
          ' seconds');
END;
/
```

The procedure displays this information:

```
Max rejected records: 150

Buffer time: 5 seconds
```

TABLE_CREATE Procedure

This procedure creates the table targets for the OLAP logs. You must have the `CREATE TABLE` privilege to use this procedure.

`TABLE_CREATE` also upgrades existing log tables to the current version while preserving the data.

Syntax

```
DBMS_CUBE_LOG.TABLE_CREATE (
          LOG_TYPE       IN   BINARY_INTEGER  DEFAULT,
          TBLNAME        IN   VARCHAR2        DEFAULT );
```

Parameters

Table 61-9 TABLE_CREATE Procedure Parameters

Parameter	Description
`log_type`	One of the following log types: • 1: `TYPE_OPERATIONS` • 2: `TYPE_REJECTED_RECORDS` • 3: `TYPE_DIMENSION_COMPILE` • 4: `TYPE_BUILD` See "Logging Types".
`tblname`	A table name for the log. These are the default names: • `CUBE_OPERATIONS_LOG` • `CUBE_REJECTED_RECORDS` • `CUBE_DIMENSION_COMPILE` • `CUBE_BUILD_LOG`

Examples

This example creates a Cube Operations log table named `CUBE_OPERATIONS_LOG`:

```
EXECUTE dbms_cube_log.table_create;
```

This example creates a Cube Rejected Records log table named `CUBE_REJECTED_RECORDS`:

```
EXECUTE dbms_cube_log.table_create(dbms_cube_log.type_rejected_records);
```

The next example creates a Cube Build log table named `MY_BUILD_LOG`:

```
EXECUTE dbms_cube_log.table_create -
     (dbms_cube_log.type_build, 'MY_BUILD_LOG');
```

TARGET_FILE Function

This function returns the integer value of a file target in `DBMS_CUBE_LOG` subprograms.

Syntax

```
DBMS_CUBE_LOG.TARGET_FILE ()
     RETURN BINARY_INTEGER;
```

Returns

```
3
```

Usage Notes

Use this function instead of its binary integer equivalent for the *`LOG_LEVEL`* parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Targets".

Example

This command disables the Cube Operations log file:

```
EXECUTE dbms_cube_log.disable -
     (dbms_cube_log.type_operations, dbms_cube_log.target_file);
```

TARGET_LOB Function

This function returns the integer value of a LOB target.

Syntax

```
DBMS_CUBE_LOG.TARGET_LOB ()
     RETURN BINARY_INTEGER;
```

Returns

```
4
```

Usage Notes

Use this function instead of its binary integer equivalent for the *`LOG_LEVEL`* parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Targets".

Example

This command disables the Cube Operations log LOB:

```
EXECUTE dbms_cube_log.disable -
     (dbms_cube_log.type_operations, dbms_cube_log.target_lob);
```

TARGET_TABLE Function

This function returns the integer value of a table target.

Syntax

```
DBMS_CUBE_LOG.TARGET_TABLE ()
     RETURN BINARY_INTEGER;
```

Returns

1

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_TARGET` parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Targets".

Example

This command disables the Cube Operations log table:

```
EXECUTE dbms_cube_log.disable -
     (dbms_cube_log.type_operations, dbms_cube_log.target_table);
```

TARGET_TRACE Function

This function returns the integer value of a trace file target.

Syntax

```
DBMS_CUBE_LOG.TARGET_TRACE ()
     RETURN BINARY_INTEGER;
```

Returns

2

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_TARGET` parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Targets".

Example

This command disables the Cube Operations log trace file:

```
EXECUTE dbms_cube_log.disable -
     (dbms_cube_log.type_operations, dbms_cube_log.target_trace);
```

TYPE_BUILD Function

This function returns the integer value of the Cube Build log.

Syntax

```
DBMS_CUBE_LOG.TYPE_BUILD ()
     RETURN BINARY_INTEGER;
```

Returns

4

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_TYPE` parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Types".

Example

This query returns the default name of a Cube Build log:

```
SELECT dbms_cube_log.default_name(dbms_cube_log.type_build) "Log Name" -
     FROM dual;

Log Name
-------------------------
CUBE_BUILD_LOG
```

TYPE_DIMENSION_COMPILE Function

This function returns the integer value of the Cube Dimension Compile log.

Syntax

```
DBMS_CUBE_LOG.TYPE_DIMENSION_COMPILE ()
     RETURN BINARY_INTEGER;
```

Returns

```
3
```

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_TYPE` parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Types".

Example

This query returns the default name of a Cube Dimension Compile log:

```
SELECT dbms_cube_log.default_name(dbms_cube_log.type_dimension_compile) -
     "Log Name" FROM dual;

Log Name
-------------------------
CUBE_DIMENSION_COMPILE
```

TYPE_OPERATIONS Function

This function returns the integer value of the Cube Operations log.

Syntax

```
DBMS_CUBE_LOG.TYPE_OPERATIONS ()
     RETURN BINARY_INTEGER;
```

Returns

```
1
```

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_TYPE` parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Types".

Example

This query returns the default name of a Cube Dimension Compile log:

```
SELECT dbms_cube_log.default_name(dbms_cube_log.type_operations) "Log Name" -
     FROM dual;

Log Name
------------------------
CUBE_OPERATIONS_LOG
```

TYPE_REJECTED_RECORDS Function

This function returns the integer value of the cube Cube Rejected Records log.

Syntax

```
DBMS_CUBE_LOG.TYPE_REJECTED_RECORDS ()
     RETURN BINARY_INTEGER;
```

Returns

```
2
```

Usage Notes

Use this function instead of its binary integer equivalent for the `LOG_TYPE` parameter in `DBMS_CUBE_LOG` subprograms. See "Logging Types".

Example

This query returns the default name of a Cube Rejected Records log:

```
SELECT dbms_cube_log.default_name(dbms_cube_log.type_rejected_records) -
     "Log Name" FROM dual;

Log Name
------------------------
CUBE_REJECTED_RECORDS
```

VERSION Function

This function returns the version number of a specific log table or the current version number of a specific log type.

Syntax

```
DBMS_CUBE_LOG.VERSION (
          LOG_TYPE       IN   BINARY_INTEGER  DEFAULT 1,
          TBLNAME        IN   VARCHAR2        DEFAULT NULL)
     RETURN BINARY_INTEGER;
```

Parameters

Table 61-10 VERSION Function Parameters

Parameter	Description
log_type	One of the following log types: • 1: TYPE_OPERATIONS • 2: TYPE_REJECTED_RECORDS • 3: TYPE_DIMENSION_COMPILE • 4: TYPE_BUILD See "Logging Types".
tblname	The name of the log table whose version is returned.

Returns

A version number

Examples

This example returns the current version of the Cube Operations log:

```
SELECT dbms_cube_log.version FROM dual;

   VERSION
----------
         2
```

This example returns the version number of an existing Cube Rejected Records log named CUBE_REJECTED_RECORDS.

```
SELECT dbms_cube_log.version(dbms_cube_log.type_rejected_records, -
     'CUBE_REJECTED_RECORDS') version FROM dual;

   VERSION
----------
         2
```