23 DBMS_APP_CONT_REPORT

This procedure generates a report of ACCHK trace collected while ACCHK_SET(TRUE).

This chapter contains the following topics:

- DBMS_APP_CONT_REPORT Security Model
- Summary of DBMS_APP_CONT_REPORT Subprograms

DBMS_APP_CONT_REPORT Security Model

Applications must have the EXECUTE privilege on the DBMS_APP_CONT_REPORT package.

Summary of DBMS_APP_CONT_REPORT Subprograms

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

Table 23-1 *DBMS_APP_CONT_REPORT Package Subprograms*

Subprogram	Description
ACCHK_REPORT Procedure	This procedure provides a report of your run. The data is derived from the acchk views.

ACCHK_REPORT Procedure

This procedure generates a report of your run. The data is derived from the acchk views.

Syntax

```
DBMS_APP_CONT_REPORT.ACCHK_REPORT (
   level        IN NUMBER    DEFAULT SUMMARY,
   service_name IN VARCHAR2  DEFAULT NULL,
   start_time   IN DATE      DEFAULT NULL,
   duration     IN NUMBER    DEFAULT NULL,
   source       IN NUMBER    DEFAULT FROM_TABLES);
```

Parameters

Table 23-2 ACCHK_REPORT Procedure Parameters

Parameter	Description
level	These are constants. • SUMMARY-summary report only (default) • WARNING-summary and warnings • FULL-summary and all workload capture, including where there are no warnings

Table 23-2 (Cont.) ACCHK_REPORT Procedure Parameters

Parameter	Description
`service_name`	The service name qualifier (Optional). The value `NULL` applies to all `service_names`.
`start_time`	Optional parameter for the start time to filter events and statistics. By default, starts at first available row in tables/trace with ACCHK.
`duration`	Optional parameter for the amount of time in seconds to filter events and statistics. By default, uses available rows in tables/traces starting from the `start_time` value. **Note:** If both `start_time` and duration are not specified, then all available rows in tables/traces are used for reporting.
`source`	Optional parameter for obtaining the statistics and events from trace files when capture was in read-only mode by providing the constant value `DBMS_APP_CONT_REPORT.FROM_TRACES`. `DBMS_APP_CONT_REPORT.FROM_TABLES` is the default value and uses the views `CDB_ACCHK_STATISTICS` and `CDB_ACCHK_EVENTS` for the report. There are two constants from `DBMS_APP_CONT_REPORT`: • `FROM_TABLES`- The procedure uses `CDB_ACCHK_STATISTICS` and `CDB_ACCHK_EVENTS` to show the report. • `FROM_TRACES`- The procedure uses traces to show the report.

Procedure Exceptions

The exceptions of `DBMS_APP_CONT_REPORT.ACCHK_REPORT` procedure are as follows:

- `20000`- Invalid report level, use `DBMS_APP_CONT_REPORT` with `SUMMARY` or `WARNING` or `FULL`.
- `20000`- Invalid duration, set a valid value.
- `20000`- Invalid source, use `DBMS_APP_CONT_REPORT` with either `FROM_TABLES` or `FROM_TRACES`.

Example

To get the recorded statistics and events starting from 10 minutes ago:

```
execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.FULL,
start_time => current_date -INTERVAL '10' MINUTE);
```

To get the recorded statistics and events starting from 10 minutes ago up to 420 seconds (7 minutes):

```
execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.FULL,
start_time  => current_date -INTERVAL '10' MINUTE, duration => 420);
```

To get the recorded statistics and events of the first 60 seconds starting from the first recorded statistic or event:

```
execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.FULL, duration
=> 60);
```

To get the report by using tables:

```
execute dbms_app_cont_report.acchk_report(source =>
dbms_app_cont_report.FROM_TABLES);
```

To get the report by using traces. Use this option when in read-only mode:

```
execute dbms_app_cont_report.acchk_report(source =>
dbms_app_cont_report.FROM_TRACES);
```

To get only the summary report:

```
execute dbms_app_cont_report.acchk_report(dbms_app_cont_report.SUMMARY);
```