



Event++: new event syntax on Oracle Database Server 11g (Doc ID 864734.1)

Modified: Aug 4, 2018

Type: BULLETIN

Status: PUBLISHED [LIMITED]

Visibility: **INTERNAL**

In this Document

[Purpose](#)
[Scope](#)
[Details](#)
[Brief overview of event++ syntax](#)
[Event_id](#)
[Event_scope and Event_filter](#)
[Action](#)
[Event_parameter](#)
[How to disable event](#)
[How to specify more than one actions against one event in event++](#)
[Foreground/background processes check event change in 11g](#)
[Using oradebug with event++ syntax](#)
[How to read online document via oradebug](#)
[References](#)

Oracle Confidential LIMITED - Distribute to customer under guidance from an Oracle Engineer.

Reason: event is used under guidance from Oracle Support[This section is not visible to customers.]

APPLIES TO:

Oracle Database - Enterprise Edition - Version 11.1.0.6 and later
 Oracle Database Cloud Schema Service - Version N/A and later
 Oracle Database Exadata Cloud Machine - Version N/A and later
 Oracle Database Exadata Express Cloud Service - Version N/A and later
 Oracle Cloud Infrastructure - Database Service - Version N/A and later
 Information in this document applies to any platform.

PURPOSE

Event++ is introduced in Oracle 11g, which rewrite event feature until 10.2. New event syntax is introduced, and also there is some change on how to specify event and how events are registered. This article shows 11g change from 10g and overview of new event ++ syntax.

SCOPE

This applies to database administrators and support engineers who use or guide events on 11g.

DETAILS

Brief overview of event++ syntax

11g event syntax, so to say, event++ syntax is shown as follows.

```
<event_spec>      ::= '<event_id> [<event_scope>]
                        [<event_filter_list>]
                        [<event_parameters>]
                        [<action_list>]
                        [off]'
```

```
<event_id>        ::= <event_name | number>[<target_parameters>]
<event_scope>     ::= [<scope_name>: scope_parameters]
<event_filter>    ::= {<filter_name>: filter_parameters}
<action>          ::= <action_name>(action_parameters)
<*_parameters>   ::= <parameter_name> = <value>[, ]
```

10g syntax does not match with above picture. But there is upper compatibility, and most 10g syntax works fine in 11g even



Event_id

Event_id controls when event is triggered or examined. In 11g, new event ids including trace[] are introduced. Generally speaking, event_id which can be used in 10g works fine in 11g.

trace[] syntax in event_id is part of UTS tracing. This article does not include usage of available event_id, as well as details of UTS tracing feature and example.

In 11g, roughly speaking we have following kind of event id.

- Event triggered when that error is signaled. e.g. 600, 7445
- Event which enables some kind of diagnostics. e.g. 10046, 10053
- Named events which change behavior, enable trace, and etc
- "immediate" which force event action run immediately
- trace[] which controls UTS tracing

Event_scope and Event_filter

Event_scope is event++ new feature, this is used to specify period when event should be fired.

Event_filter is also event++ new feature, this is used to specify condition when this event should be fired.

For details of event_scope and event_filter, please see unpublished note:869223.1

Action

Action is "errorstack", "heapdump", and so on. This is similar with 10g, but following points are different

- In 10g, we specify event level by "<action name> level <level>" syntax
In event++, we use "<action name> (<level>)" syntax

```
event = "942 trace name errorstack forever, level 1" -- 10g syntax
event = "942 errorstack(1)" -- event++ syntax
```

- With 10g syntax, unless we set 'forever' explicitly event is fired only once. But with event++ syntax, event is triggered every time. If you need to make event be fired only once, use event_filter "occurrence" as follows.

```
SQL> alter session set events '942 {occurrence:end_after 1} errorstack(1)';
```

Please note that until [bug 16050820](#) is resolved it will be necessary to spell the word 'occurrence' incorrectly. Therefore the syntax will be

```
SQL> alter session set events '942 {occurence:end_after 1} errorstack(1)';
```

```
SQL> alter session set events '942 errorstack(1)';
```

----- (1) event++ syntax

```
SQL> alter session set events '942 trace name errorstack level 1';
```

----- (2) 10g syntax

```
SQL> alter session set events '942 trace name errorstack forever, level 1';
```

----- (3) 10g syntax

Raise ORA-942 error twice like follows. You will get errorstack for 2nd statement by (1) and (3) setting above.

```
SQL> drop table non_existent_table_1;
ERROR at line 1:
ORA-00942: table or view does not exist
```


```
SQL> drop table from non_existent_table_2;
ERROR at line 1:
ORA-00942: table or view does not exist
```

Event_parameter

Most event accept level as event parameter. when you don't specify level explicitly, 1 is used as default.

```
SQL> alter session set events 'sql_trace level 1';
is same with
SQL> alter session set events 'sql_trace';
```

To specify multiple parameters in event_parameters, use comma separated list. And how to specify multiple parameters for other *_parameters is same.



```

To specify multiple event_parameters for sql_trace event.
SQL> alter session set events 'sql_trace wait=true, bind=true';

You can omit '=' and following works fine.
SQL> alter session set events 'sql_trace wait true, bind true';

Example to specify multiple action_parameter for cursortrace action.
SQL> alter session set events 'immediate cursortrace(level=99172, address=1745700775)';

```

How to disable event

To disable event, we use "off" like 10g syntax. You need to supply event_id to disable.

```
SQL> alter session set events 'sql_trace off';
```

How to specify more than one actions against one event in event++

Until 10.2, we can specify multiple actions to an event with following method.

- init parameter file

```

* init parameter file -- 10g example
event = "942 trace name errorstack level 1"
event = "942 trace name systemstate level 1"
* alter system/session -- 10g example
alter system set events '942 trace name errorstack level 1';
alter system set events '942 trace name systemstate level 1';

```

In above example, we set errorstack and systemstate event to be triggered by ORA-942 error.

From 11g, if we use above way, Oracle overwrites previous action set for same event and thus only one action is enabled. To avoid this behavior and enable more than one action, use following new event syntax.

```

* init parameter file -- event++ example
event = "942 errorstack(1) systemstate(1)"
* alter system/session -- event++ example
alter system set events '942 errorstack(1) systemstate(1)';

```

If you want to keep 10g syntax, following works as well in 11g also.

```

event = "942 trace name errorstack level 1; name systemstate level 1"
alter system set events '942 trace name errorstack level 1; name systemstate level 1';

```

Foreground/background processes check event change in 11g

Until 10.2, even we set event at system level like

```
SQL> alter system set events '10046 trace name context forever, level 1';
```

This event is effective only for new sessions started after this change.

But in 11g, foreground and background processes periodically check event change and those events affect existing processes soon.

This applies if process works well and is not waiting/spinning. So if target process is waiting for a long time (like "SQL*Net message from client", "smon timer") or spinning due to some problem, those processes won't notice event change until exiting from waiting/spinning status. For most wait events, we wake up periodically with timeout and check if waiting resource is available or not. and if waiting resource is still unavailable, it goes into same wait again. So even process is waiting, it will notice event change when it wakes up. But if something wrong and process does not wake up from wait event, or that wait event does not have timeout (like "SQL*Net message from client"), process does not notice wait event change.

Thus, if you want to set event for spinning process or process who does not wake up from event at all, use oradebug instead.

Using oradebug with event++ syntax

oradebug also accepts event++ syntax.

```

SQL> oradebug event sql_trace wait=true, bind=true
SQL> oradebug event sql_trace off

```



How to read online document via oradebug

Using oradebug, you can see short help for event++ syntax. To see it,

```
SQL> connect / as sysdba
SQL> oradebug setmypid
SQL> oradebug doc event
```

More help document is available, try "oradebug doc event name", "oradebug doc event action", "oradebug doc event name sql_trace", and so on.

Code Reference

event definition: DBGD_EVENT / DBGFILCSD_DBGD_EVENT

action definition: DBGFILCSD_DBGD_ACTION

scope definition: DBGD_SCOPE

filter definition: DBGD_POSTFILTER

parameter definition: DBGD_PARAM_DEF

[This section is not visible to customers.]

REFERENCES

- [BUG:8593858](#) - DOES NOT ENABLE THE SETTING OF MULTIPLE EVENTS USING SPFILE[This section is not visible to customers.]
- [NOTE:869223.1](#) - Event++: summary for event scope and event filter[This section is not visible to customers.]
- [BUG:16050820](#) - MISSPELLING OF KEYWORD "OCCURRENCE"[This section is not visible to customers.]
- [NOTE:1567533.1](#) - Event++: 12c new feature and example[This section is not visible to customers.]

Didn't find what you are looking for?

Document Attributes

Author:	TFUJIKAW.JP;RCITTON.IT;	Status:	PUBLISHED(LIMITED)
Owner:	TFUJIKAW.JP	Publisher:	RSOFRONI.RO
Alias:		Content Type:	TEXT/X-HTML
Distribution:	LIMITED	Visibility:	INTERNAL
Created By:	TFUJIKAW.JP	Created:	Jul 29, 2009
Modified By:	XIUWLI.CN	Modified:	Nov 27, 2019
Reviewed By:	JSOENDER.DK	Reviewed:	Apr 25, 2012
Source:	AWIZ	Exception:	No
Priority:	3		