WHITEPAPER

SPINNAKER SUPPORT

How to Handle ORA-00600

Table of Contents

How to Handle an ORA-00600

DEFINITION:

The ORA-00600 error, a generic internal error number for Oracle program exceptions, indicates that a process has encountered a low-level, unexpected condition. This can potentially disrupt your system and should be addressed promptly.

The ORA-00600 error includes a list of arguments in square brackets: ORA-00600 "internal error code, arguments: [%s], [%s],[%s],[%s],[%s]"

The first argument is the internal message number or character string. This argument and the database version number are critical in identifying the root cause and its potential impact on your system. The remaining arguments in the ORA-00600 error text supply further information (e.g., values of internal variables, etc.).

The first argument may help to narrow your problem to known issues. However, this argument can point to functionality that is referenced by many areas in the Oracle source code. The more detailed call stack information reported in the ORA-00600 error trace may be needed to find a solution.

Background on ORA-00600

COMMON CAUSES OF ORA-00600:

Here are some of the reasons for getting this error:

1. **File Corruption:** When an Oracle-supported file has been corrupted, you may get the ORA-00600 internal error code.
2. **Timeouts:** If a timeout occurs while connecting to the database or the query runs for a long time and timeout occurs, this internal error code occurs.
3. **Failed Data Checks in the Memory:** The ORA-00600 internal error code occurs due to failed data checks in the memory.
4. **Hardware/Memory/IO:** This error may occur when the hardware is incompatible or the memory is full in the Oracle version, and you are still working on it.
5. **Incorrectly Restored Files:** This error will occur when the Oracle files are restored improperly.

There are several reasons for the ORA-00600 error, like compatibility issues and database block errors. Sometimes, this kind of error remains unsolvable. However, it is quite difficult to find the root cause of this error.

HOW TO WORK WITH ORA-00600

Our technical team, with an average of 19+ years of experience in Oracle Technology, will work closely with the customer team as a whole to investigate the root cause of the ORA-00600 errors. Their role is crucial in identifying and resolving these issues.

Our resolution process for ORA-00600 errors is thorough and comprehensive, ensuring that all potential causes are considered and addressed.

We need to identify the root cause using the error log files, and the tools present in the system, for the common errors listed above, for ex:

- Alert_<instname>.log –
  Details all the events in the database
  - With the information we collect from the alert log, we can identify the trace file with the error handling.
- Convert the trace file using the system's tkprof utility to analyse the internal operations active during the ORA-00600 event.
- Using ASH reports to make sure we collect all the information about active sessions at the time of the event
- List the detailed operations which are active at the time of the event, ex: backup running, huge IO operations, RBS activity

These are only examples of the steps to investigate the root cause; they're not the exhaustive list.

EXAMPLE ONE ORA-00600

ORA-00600: internal error code, arguments: [kccsbck_first], [1], [3978145973]

CAUSE:

This is the generic internal error number for Oracle program exceptions. This indicates that a process has encountered an exceptional or unexpected condition. Although it often suggests a bug, ORA-00600 can result from many other reasons: timeouts, limited system resources (e.g. memory and disk space), OS issues (e.g. caching and memory management), server crashes, network failures, power failures, disk drive crashes, I/O errors, incorrectly archived files and even driver issues. In many cases, ORA-00600 denotes an existing file corruption or a data check failure in Oracle RDBMS.

ACTION:

CASE 1

```
C:\>set ORACLE_SID=MOMPRD
C:\>sqlplus sys/XXXXX@MOMPRD as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Sat Jan 29 12:28:00 2011
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> startup

ORACLE instance started.
Total System Global Area  3423965184 bytes
Fixed Size                2180544 bytes
Variable Size             2080377408 bytes
Database Buffers          1325400064 bytes
Redo Buffers              16007168 bytes

ORA-00600: internal error code, arguments: [kccsbck_first], [1], [3978145973],
[], [], [], [], [], [], [], [], []
```

SOLUTION:

After getting the ORA-00600, shut the database down, start it in the mount stage, and then alter the database to the open stage.

```
SQL> shut immediate
ORA-01507: database not mounted
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.
Total System Global Area 3423965184 bytes
Fixed Size                2180544 bytes
Variable Size             2080377408 bytes
Database Buffers          1325400064 bytes
Redo Buffers              16007168 bytes
Database mounted.

SQL> alter database open;
Database altered.

SQL> select name,open_mode from v$database;
NAME   OPEN_MODE
------ ----------
MOMPRD READ WRITE
```

CASE 2

As the above but a different approach.

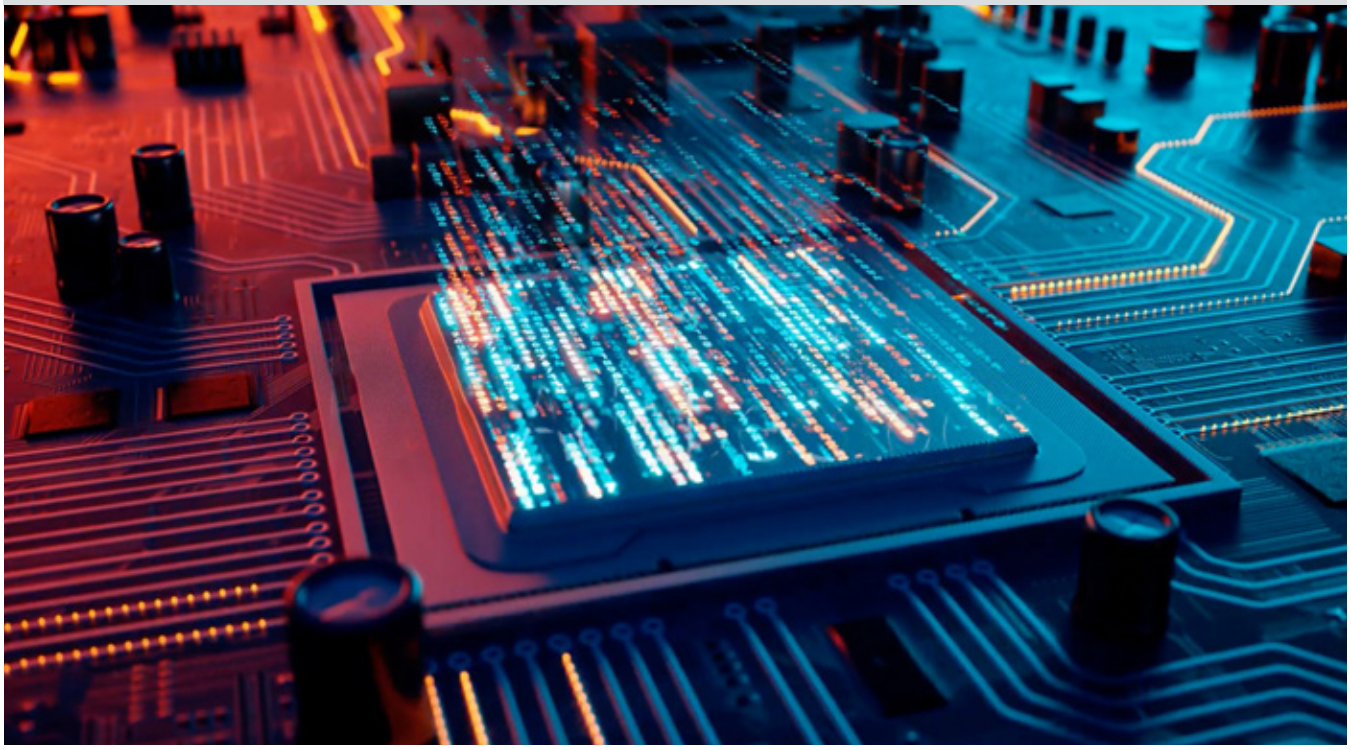
SOLUTION:

Despite the initial approach not yielding the desired results, we remained committed to finding a solution for the ORA-00600 error, which ultimately led us to a successful resolution.

```
SQL> shut immediate
ORA-01507: database not mounted
ORACLE instance shut down.

C:\>set ORACLE_SID= MOMPRD
C:\>sqlplus sys/xxxxxxxx@MOMPRD  as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Sat Jan 29 12:28:00 2011
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Connected to an idle instance.

SQL> startup mount
ORACLE instance started.
Total System Global Area 3423965184 bytes
Fixed Size                2180544 bytes
Variable Size             2080377408 bytes
Database Buffers          1325400064 bytes
Redo Buffers              16007168 bytes
Database mounted.
SQL> alter system flush shared_pool;
SQL> alter database open
Database altered.
SQL> select name,open_mode from v$database;
NAME         OPEN_MODE
---------    ----------
MOMPRD       READ WRITE
```



EXAMPLE TWO ORA-00600

ORA 600 [KGL-heap-size-exceeded]

Time ago, a web page was rendered blank for a specific group of users.

The errors were coming from an Oracle instance. One SysAdmin restarted the instance, but the errors continued.

Often there are problems due to having two different worlds: Development and Production/Operations.

What works in Development, or even in Docker, may not work at Scale in Production.

That query that works with 100,000 products, may not work with 10,000,000.

We have programmed a lot for the web, so when we saw a blank page, we knew it was an internal error, as the headers sent by the Web Server indicated 500. DBAs were seeing an elevated number of errors in one of the Servers.

So we went straight to the Oracle's logs for that Servers.

```
cat /u01/app/oracle/diag/rdbms/world7c/world7c2/alert/log.xml | grep "ERR" -B4 -A3
```

This returned several errors of the kind ORA 600 [ipc_recreate_que_2]" but this was not the error our bad guy was:

'ORA 600 [KGL-HEAP-SIZE-EXCEEDED]'

The XML fragment was similar to this:

```
<msg time='2016-01-24T13:28:33.263+00:00' org_id='oracle' comp_id='rdbms'
msg_id='7725874800' type='INCIDENT_ERROR' group='Generic Internal Error'
level='1' host_id='gotham.world7c.justice.cat' host_addr='10.100.100.30'
pid='281279' prob_key='ORA 600 [KGL-heap-size-exceeded]' downstream_comp='LIBCACHE'
errid='726175' detail_path='/u01/app/oracle/diag/rdbms/world7c/world7c2/trace/world7c2_ora_281279.trc'>
<txt>Errors in file /u01/app/oracle/diag/rdbms/world7c/world7c2/trace/world7c2_ora_281279.trc  (incident=726175):
ORA-00600: internal error code, arguments: [KGL-heap-size-exceeded], [0x14D22C0C30], [0], [524288008], [], [], [], [],
[], [], [], []
</txt></msg>
```

Just before this, there was an error with a Query, and the PID matched, so it seemed clear that the query was causing the crash at the Oracle level.

Checking the file:

```
/u01/app/oracle/diag/rdbms/world7c/world7c2/trace/world7c2_ora_281279.trc
<msg time='2016-01-24T13:28:33.263+00:00' org_id='oracle' comp_id='rdbms'
msg_id='7725874800' type='INCIDENT_ERROR' group='Generic Internal Error'
level='1' host_id='gotham.world7c.justice.cat' host_addr='10.100.100.30'
pid='281279' prob_key='ORA 600 [KGL-heap-size-exceeded]' downstream_comp='LIBCACHE'
errid='726175' detail_path='/u01/app/oracle/diag/rdbms/world7c/world7c2/trace/world7c2_ora_281279.trc'>
<txt>Errors in file /u01/app/oracle/diag/rdbms/world7c/world7c2/trace/world7c2_ora_281279.trc  (incident=726175):
ORA-00600: internal error code, arguments: [KGL-heap-size-exceeded], [0x14D22C0C30], [0], [524288008], [], [], [], [],
[], [], [], []
</txt> </msg>
```

In our case, the query launched by the Back End was using more memory than allowed, which caused Oracle to kill it.

That is a tunable that you can modify introduced in Oracle 10g.

You can see the current values first:

```
SQL> select
2 nam.ksppinm NAME,
3 nam.ksppdesc DESCRIPTION,
4 val.KSPPSTVL
5 from
6 x$ksppi nam,
7 x$ksppsv val
8 where nam.indx = val.indx and nam.ksppinm like '%kgl_large_heap_%_threshold%';

NAME                                | DESCRIPTION                                  | KSPPSTVL
=============================================================================================
_kgl_large_heap_warning_threshold     maximum heap size before KGL                   4194304
                                      writes warnings to the alert log

_kgl_large_heap_assert_threshold      maximum heap size before KGL                   4194304
                                      raises an internal error
```

So, **_kgl_large_heap_warning_threshold** is the maximum heap before getting a warning, and **_kgl_large_heap_assert_threshold** is the maximum heap before getting the error.

Depending on your case, the solution can be either:

- Breaking your query into several to reduce the memory used
- Use paginating or **LIMIT**
- Set a more significant value for those tunnables.

Setting 0 for these variables will work, although I don't recommend it, as you want your Server to kill queries that are taking more memory than you want.

To increase the value, you have to update it. Please note it is in bytes, so for 32MB is 32 * 1024 * 1024, so 33,554,432, and using spfile:

```
SQL> alter system set "_kgl_large_heap_warning_threshold"=33554432
scope=spfile ;

SQL> shutdown immediate

SQL> startup

SQL> show parameter _kgl_large_heap_warning_threshold

NAME                                 | TYPE                 | VALUE
=============================================================================================
_kgl_large_heap_warning_threshold    integer                33554432
```

Or if using the parameter file, set:

```
_kgl_large_heap_warning_threshold=33554432
```

SPINNAKER SUPPORT PROCESS

DATABASE BUGS CAN'T BE ADDRESSED BY ANYONE OTHER THAN ORACLE.

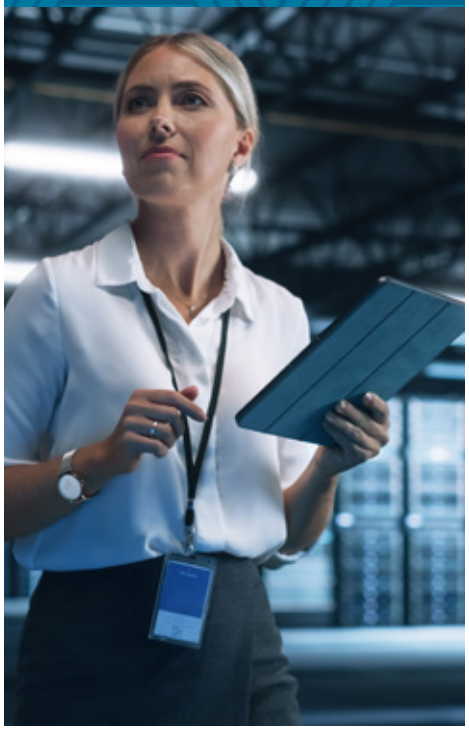
Let's begin with a basic understanding of the difference between the two support models. As with any issue, there is always more than one way to solve the problem. What Oracle does—applying a patch at the binary code level—is certainly one of them. This takes time to develop and test and may or may not impact your application customisations.

Our engineers also review issues for the source and context of the problem. After considering the most applicable approach, they provide an equal or superior solution based on how you are using the existing technology. That can either be a patch from your archive – created for you during the onboarding process – or a software reconfiguration tailored to your specific needs and system requirements.

So, this question is misleading. As the software provider, Oracle is the only vendor that can access the Oracle product's source code and provide changes at the base or binary code level of the software. However, they are not the only vendor that can perform root cause analysis of a newly discovered issue or vulnerability and provide a resolution.

For example, with break/fix type issues in databases and other technology products, we triage the issue to understand exactly why it is occurring. Once we understand the exact source, we can develop a solution that prevents the defective code from being executed, effectively bypassing the bug or defect. The solution could be a parameter change, a change to application code (where possible), or even an execution plan change to control the issue.

Our engineers also review issues for the source and context of the problem. After considering the most applicable approach, they provide an equal or superior solution based on how you are using the existing technology.



SPINNAKER SUPPORT PROCESS (CONT.)

WITHOUT ACCESS TO SOURCE CODE, WE CAN'T POSSIBLY DEVELOP A FIX.

Spinnaker Support follows industry standards and best practice customization standards when developing code fixes for our customers and documents any code fixes within the customer's system.

Spinnaker Support's support engineers will work to identify the root cause of the issue within the code and, if necessary, develop a specific code fix for our customer's system. They will also support the deployment or migration of these fixes through our client's regression and move to production testing.

In addition to following internal system best practices for documentation of code changes, these changes are also documented within Spinnaker Support's ticketing system for future reference.

WITHOUT ACCESS TO COMPILED, WE CAN'T POSSIBLY DEVELOP A FIX.

Since our Company's inception 12 years ago, Spinnaker Support has provided fixes for thousands of tickets (>14,835 in the past year alone) for over 1,280 customers.

None of these fixes required access to the source code, as Spinnaker Support develops fixes that work around the source code. Spinnaker Support's approach to all tickets is to start with root-cause analysis. When we understand the nature of the issue, we develop the fix and put it through our QA process. Then, we ship it to our customer, who does their own QA and transports it into production.

Of interest, unlike Oracle, Spinnaker Support also fixes issues with customized code, and this additional benefit is included in our pricing. I would be pleased to include this topic in our forthcoming technical team call.

FIXES AND PATCHES PROVIDED BY THIRD-PARTY SUPPORT PROVIDERS INTRODUCE EXCESSIVE CUSTOMIZATIONS AND HINDER THE PATH TO A FUTURE UPGRADE.

The opposite is true. Spinnaker Support adheres to ITIL and the specific software vendor's standards for developing any solution. Any customizations developed to resolve an issue meet the software manufacturer's specifications and development standards and will in no way hinder a company's ability to upgrade its software in the future.