Lock Request Timeout Error 1222 State 51

Last updated by | Ricardo Marques | Feb 16, 2023 at 2:50 AM PST

Contents

Issue

Users receiving lock timeout request (Error 1222, State 51).

```
Failed to retrieve data for this request. (Microsoft.SqlServer.Management.Sdk.Sfc)
Lock request time out period exceeded. (Microsoft SQL Server, Error: 1222)
```

Investigation/Analysis

Lock timeout requests error 1222 indicates that another transaction held a lock on a required resource longer than this query could wait for it.

From customer side

Perform the following tasks to alleviate the problem:

- Locate the transaction that is holding the lock on the required resource, if possible. Use sys.dm_os_waiting_tasks and sys.dm_tran_locks dynamic management views.
- If the transaction is still holding the lock, terminate that transaction if appropriate.
- Execute the query again.

If this error occurs frequently change the lock time-out period or modify the offending transactions so that they hold the lock for less time.

Investigate from our side

Check for timeout errors

Confirm the timeout errors for the server at the timeframe mentioned:

```
AlrSQLErrorsReported
| where TIMESTAMP between ({StartDateTime}..{EndDateTime})
| where LogicalServerName == "{MIName}" //Input customer server name
| where error_number == '1222'
| project originalEventTimestamp, NodeName, database_name, ['state'], code_package_version
| summarize count() by bin(originalEventTimestamp, 1h), database_name
| render timechart
```

Check for any login outages at the time of issue

```
LoginOutages
| where TIMESTAMP between ({StartDateTime}..{EndDateTime})
| where logical_server_name == "{MIName}" //Input customer server name
```

If there were login outages at the time of issue, further investigation is needed from an availability perspective.

Check if there were any connectivity issues

```
MonLogin
| where TIMESTAMP between ({StartDateTime}..{EndDateTime})
| where logical_server_name == "{MIName}" //Input customer server name
| where event == "process_login_finish"   and package == "sqlserver"
| where is_user_error == 0
| extend success=case(is_success == 1, 1, 0)
| extend failure=case(is_success == 0, 1, 0)
| extend failure_enqueue_time_ms=case(is_success == 1, total_time_ms, 0)
| summarize Successful_Login_Count=sum(success), Failed_Login_Count=sum(failure) by bin(originalEventTimestamp
| render timechart
```

If there were login outages at the time of issue, further investigation is needed from an availability perspective.

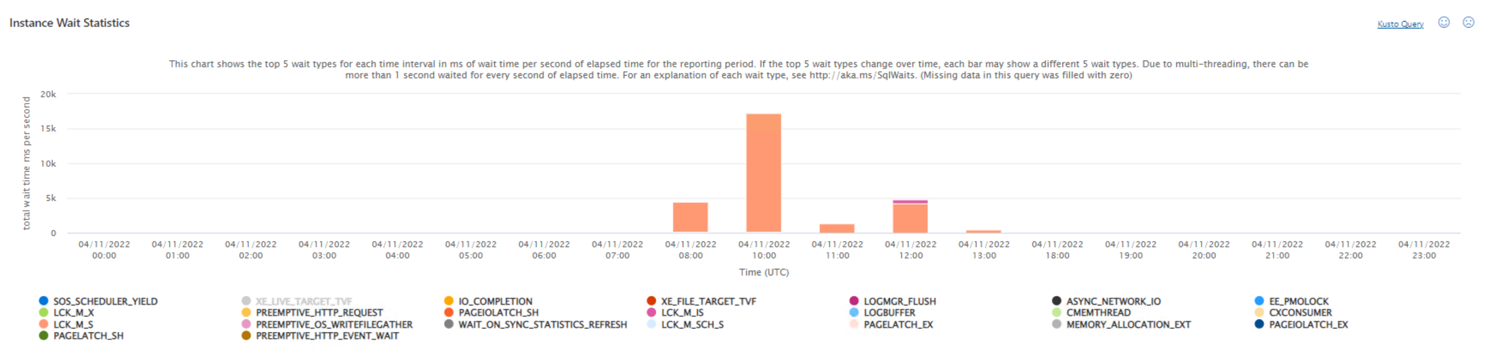
Check if customer workload increased at the time of issue

Check total query execution count vs error query count and timeout query count to see if the errors occurred during high customer workload

```
MonWiQdsExecStats
| where TIMESTAMP between ({StartDateTime}..{EndDateTime})
| where LogicalServerName =~ "{MIName}" //Input customer server name
| summarize total_query_execution_count = sum(execution_count)
    , timeout_query_count = sumif(execution_count, exec_type == 3)
    , error_query_count = sumif(execution_count, exec_type == 4)
    by bin(originalEventTimestamp, 15m)
| sort by originalEventTimestamp asc nulls last
| render timechart
```

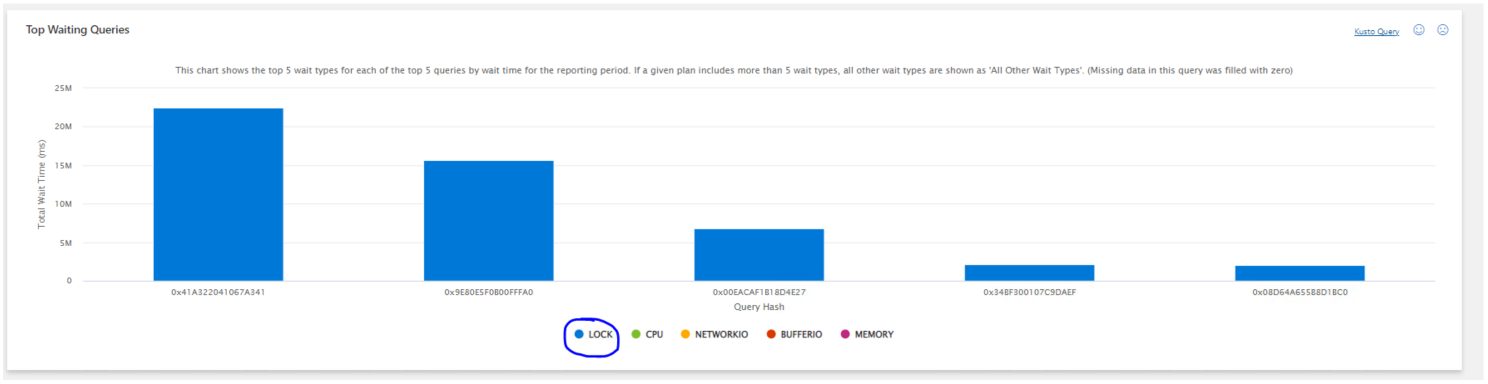
Check ASC for locks

Check ASC --> Performance --> Overview tab to see if there dominant wait type of LCK_*


Instance Wait Statistics

This chart shows the top 5 wait types for each time interval in ms of wait time per second of elapsed time for the reporting period. If the top 5 wait types change over time, each bar may show a different 5 wait types. Due to multi-threading, there can be more than 1 second waited for every second of elapsed time. For an explanation of each wait type, see http://aka.ms/SqlWaits. (Missing data in this query was filled with zero)

Then Check ASC --> Performance --> Queries tab to check for queries that are waiting on the lock resource.


Top Waiting Queries

This chart shows the top 5 wait types for each of the top 5 queries by wait time for the reporting period. If a given plan includes more than 5 wait types, all other wait types are shown as 'All Other Wait Types'. (Missing data in this query was filled with zero)

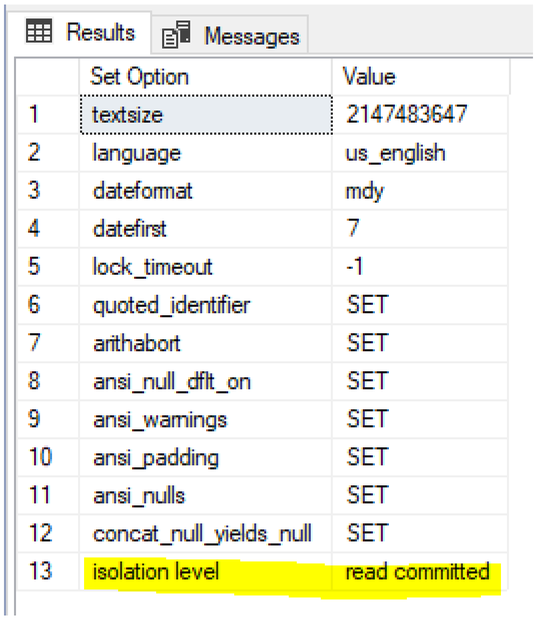
Once you identify the dominant wait type as LCK_* and the queries waiting on lock, check query type (select, delete, update, insert) and check for lock compatibility

For example, if the query type is Select and they are waiting on LCK_M_S, implies that customer is not running with read committed snapshot isolation level. In read committed isolation, readers will be blocked by writers, which can cause heavy locking and lock timeouts, which is not the case with read committed snapshot isolation level. General recommendation is to consider switching to read committed snapshot isolation level (it is default isolation level in Azure SQL and will result in a state where readers won't be blocked on writers. However, READ_COMMITTED_SNAPSHOT is **not** default on Azure SQL Managed Instance).

Customers can check isolation level with these two queries:

- This will tell exact isolation level:

```
DBCC USEROPTIONS
```

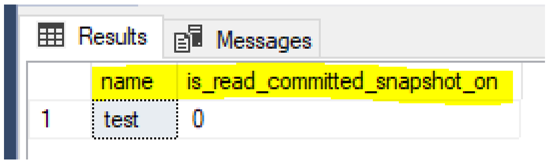


Results Messages

	Set Option	Value
1	textsize	2147483647
2	language	us_english
3	dateformat	mdy
4	datefirst	7
5	lock_timeout	-1
6	quoted_identifier	SET
7	arithabort	SET
8	ansi_null_dflt_on	SET
9	ansi_warnings	SET
10	ansi_padding	SET
11	ansi_nulls	SET
12	concat_null_yields_null	SET
13	isolation level	read committed

- In case of read committed isolation level, snapshotting option can be enabled or disabled. This query will tell if read committed with snapshot is enabled.

```
SELECT name, is_read_committed_snapshot_on
FROM sys.databases
WHERE name = DB_NAME();
```



Results Messages

	name	is_read_committed_snapshot_on
1	test	0

- Customers can switch to read committed snapshot by executing this query:

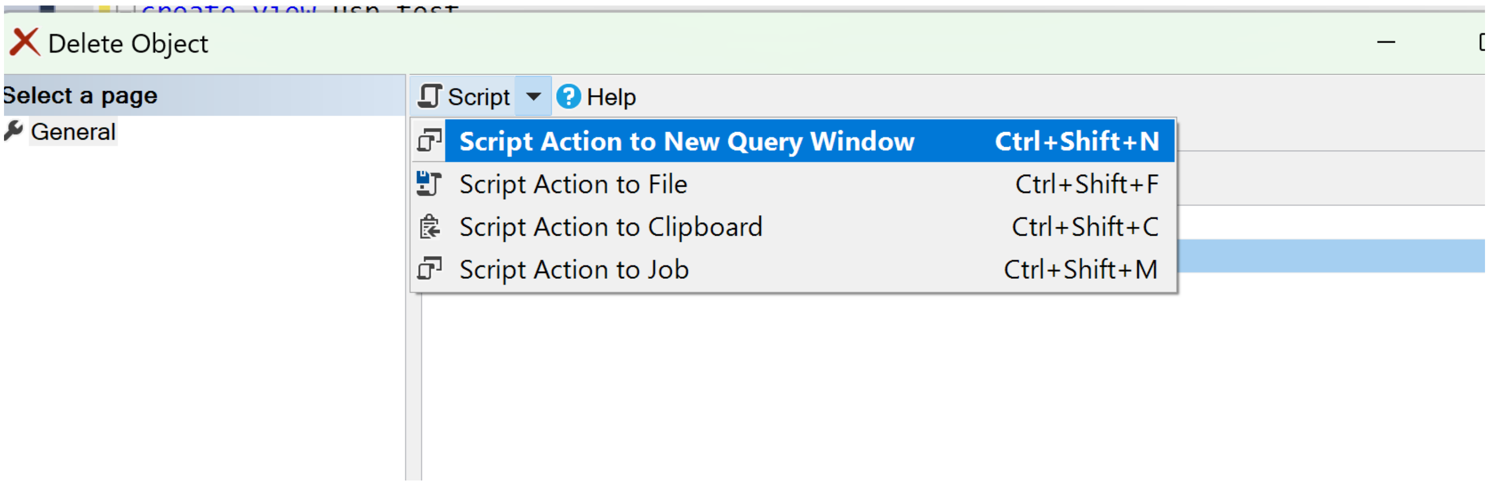
```
ALTER DATABASE <database_name> SET READ_COMMITTED_SNAPSHOT ON
```

Customer can use queries from above to verify that read committed snapshot is now on (column is_read_committed_snapshot_on should be 1).

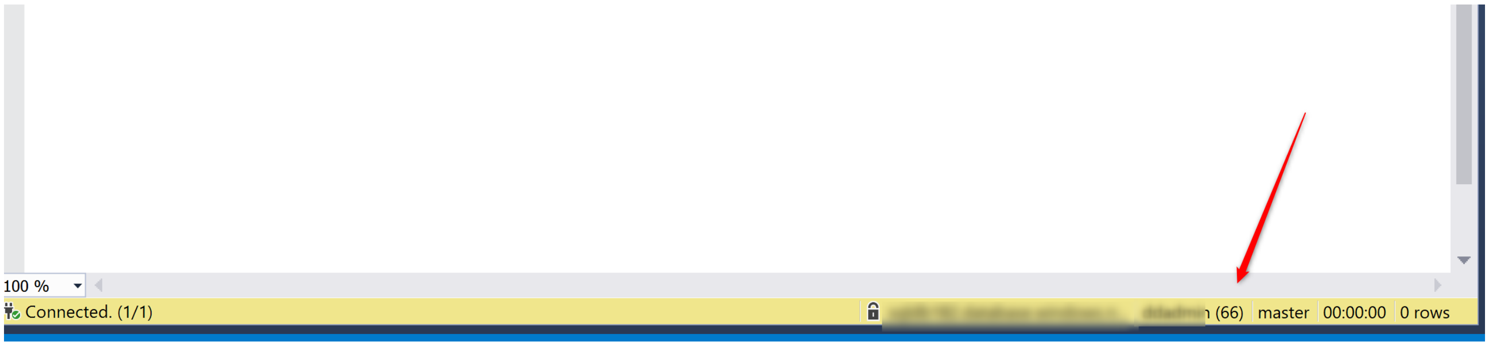
Besides of ASC, if the issue is reproducible, check if there is blocking on the customer environment. If the error appears when the customer tries some kind of operation on the SSMS UI, you might want to script the operation since you will have more control on it.

Example:

- the customer is dropping a view. Write the SQL or just generate the code:


Delete Object
Select a page
General
Script
Help
Script Action to New Query Window Ctrl+Shift+N
Script Action to File Ctrl+Shift+F
Script Action to Clipboard Ctrl+Shift+C
Script Action to Job Ctrl+Shift+M

- execute from SSMS. Note that you will have the SPID number information.



- check for the existence of blocking (while the operation is running) using the query below

```sql
select
    des.session_id
    ,des.status
    ,des.login_name
    ,des.[host_name]
    ,der.blocking_session_id
    ,db_name(der.database_id) as database_name
    ,der.command
    ,des.cpu_time
    ,des.reads
    ,des.writes
    ,dec.last_write
    ,des.[program_name]
    ,der.wait_type
    ,der.wait_time
    ,der.last_wait_type
    ,der.wait_resource
    ,(
        case des.transaction_isolation_level
            when 0 then 'Unspecified'
            when 1 then 'ReadUncommitted'
            when 2 then 'ReadCommitted'
            when 3 then 'Repeatable'
            when 4 then 'Serializable'
            when 5 then 'Snapshot'
        end
     ) as transaction_isolation_level
    ,object_name(dest.objectid, der.database_id) as object_name
    ,substring(dest.text, der.statement_start_offset / 2, (
           case
                when der.statement_end_offset = - 1 then datalength(dest.text)
                else der.statement_end_offset
            end - der.statement_start_offset
        ) / 2) as [executing statement]
    ,deqp.query_plan
from
    sys.dm_exec_sessions des
    left join
        sys.dm_exec_requests der
        on des.session_id = der.session_id
    left join     sys.dm_exec_connections dec
        on des.session_id = dec.session_id
    cross apply
        sys.dm_exec_sql_text(der.sql_handle) dest
    cross apply
        sys.dm_exec_query_plan(der.plan_handle) deqp
where des.session_id = 66 --spid number used by the operation
;
```

Mitigation

- Locate the transaction that is holding the lock on the required resource, if possible.
- If the transaction is still holding the lock, terminate that transaction if appropriate.
- Execute the query again.

If this error occurs frequently change the lock time-out period or modify the offending transactions so that they hold the lock for less time. Check for increased customer workload and ensure that the queries with incompatible locks are not running for too long in parallel.

Public Doc Reference

- Error 1222

- Lock timeout
- Lock escalation
- Lock compatibility

Internal Reference

- ICM 301654797

- ICM 366964607

Root Cause Classification

Cases resolved by this TSG should be coded to the following root cause: Performance/Waits/Other

How good have you found this content?