Error 40532 State 4 on master database

Last updated by | Holger Linke | Mar 23, 2023 at 10:40 AM PDT

Contents

Issue

The customer is trying to connect to their Azure SQL Database server, but it is consistently failing with the following errors:

Example error as reported by SSMS:

> Error 40532 State 1
> Cannot connect to servername.database.windows.net
> Cannot open server 'servername' requested by the login. The login failed. (.NET SqlClient Data Provider)

Example error as reported by the portal's Query Editor:

> Error 40613
> Database 'databasename' on server 'servername' is not currently available. Please retry the connection later. If the problem persists, contact support and provide the session tracing ID of 'CA411FC8-2866-463B-81A2-E1458BF9E612'.

The errors persist with either using SQL authentication or AAD authentication.

Investigation / Analysis

> **Note:** The symptoms are similar to the issue described in Wiki article "Error 40532 State 4". The main difference appears to be that the error 40532 state 4 is reported for either the `master` database or the user database - but this is still uncertain for now due to the lack of comparable case scenarios. Make sure to check both articles for narrowing down the cause of your customer's issue.

1. MonLogin overview

The first step in the investigation is to gather the login errors from `MonLogin`:

```
let srv = "servername";
let db  = "databasename";
let startTime = datetime(2023-03-22 00:00:00Z);
let endTime   = datetime(2023-03-22 23:00:00Z);
MonLogin
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| filter logical_server_name =~ srv
| where database_name =~ "master" or database_name =~ db
| extend logical_server_name = tolower(logical_server_name), database_name = tolower(database_name)
| where event == "process_login_finish"
| where is_success == 0
| summarize
    count(),
    min_TIMESTAMP = format_datetime(min(TIMESTAMP), "yyyy-MM-dd HH:mm:ss"),
    max_TIMESTAMP = format_datetime(max(TIMESTAMP), "yyyy-MM-dd HH:mm:ss")
    by database_name , error, ['state'], peer_address , is_user_error, is_vnet_address, application_name, driv
| order by database_name, error asc, ['state'] asc
```

Sample output (abbreviated):

database_name	error	state	peer_address	is_user_error	driver_name	alias_name
databasename	33155	1	81.59.97.x	TRUE	.Net SqlClient Data Provider	
databasename	40613	126	40.127.145.x	FALSE	Core Microsoft SqlClient Data Provider	
databasename	40613	126	<IPv6 address>	TRUE	Core Microsoft SqlClient Data Provider	
master	40532	4	52.236.185.x	TRUE	ODBC	
master	40532	4	52.236.185.x	TRUE	ODBC	
master	40532	4	81.59.97.x	TRUE	.Net SqlClient Data Provider	
master	40532	4	81.59.97.x	TRUE	.Net SqlClient Data Provider	
master	40532	4	81.59.97.x	TRUE	.Net SqlClient Data Provider	
master	40532	4	81.59.97.x	TRUE	TDSSQLTestClient	
master	40607	135	131.107.160.x	TRUE	.Net SqlClient Data Provider	
master	40615	130	131.107.160.x	TRUE	.Net SqlClient Data Provider	

The significant pattern is that:

- Connections through the driver "Core Microsoft SqlClient Data Provider" are failing with error 40613 state 126 against the user database. These come from "sqlserver" and are accounting for the unavailability errors.
- Failures that are logged for drivers "ODBC" and ".Net SqlClient Data Provider" are reported against the `master` database as error 40532 state 4 from "xdbgateway". They are state=4 and not state=1 as reported to the users. These represent the vast majority of failures in this example - the actual frequency might be different in your case.

2. Tracing ID and connection_id

Some of the customer error messages might contain a "tracing ID". This aligns with MonLogin's `connection_id` column, and by querying that, you can see the details about the corresponding login attempt. Note though that the `connection_id` doesn't cover all login steps and that you have to run the following query twice: once with the tracing ID/connection_id to retrieve the `connection_peer_id`, then a second time to search for entries with that connection_peer_id:

```
let startTime = datetime(2023-03-22 00:00:00Z);
let endTime   = datetime(2023-03-22 23:00:00Z);
let timeRange = ago(1d);
MonLogin
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where connection_id =~ 'CA411FC8-2866-463B-81A2-E1458BF9E612' //or connection_peer_id =~ "888621F8-B910-4AED
| project originalEventTimestamp, NodeName, AppName, package, is_success, error, state, logical_server_name, L

Sample output:

originalEventTimestamp       NodeName  AppName       package    is_success  error   state  logical_server_name
---------------------------- --------- ------------- ---------- ----------- ------- ------ -------------------------
2023-03-22 11:44:50.0825720  DB.48     Worker        xdbhost    TRUE        0       0      servername
2023-03-22 11:44:50.0832003  DB.48     a6c044077814  sqlazure                           servername
2023-03-22 11:44:50.0832478  DB.48     a6c044077814  sqlserver
2023-03-22 11:44:50.0836025  DB.48     a6c044077814  sqlserver              -1      4
2023-03-22 11:44:50.0836060  DB.48     a6c044077814  sqlserver              -1      3
2023-03-22 11:44:50.0837328  DB.48     a6c044077814  sqlserver
2023-03-22 11:44:50.0838147  DB.48     a6c044077814  sqlserver              -1      4
2023-03-22 11:44:50.0838168  DB.48     a6c044077814  sqlserver              -1      3
2023-03-22 11:44:50.1484915  DB.48     a6c044077814  sqlazure               40613           servername
2023-03-22 11:44:50.1485112  DB.48     a6c044077814  sqlserver                      84
```

Note how this data doesn't include any "process_login_finish" event. The login succeeds with the xdbhost at the gateway, but is then failing with error 40613 state 84 while apparently reaching out to the actual user database. This error is raised, however, when logins fail because of a failure to connect to the logical master.

3. Cross-check server names for internal DNS and routing

From the results on #1 and #2 above, there is a possible hint towards a discrepancy in the routing:

- if we connect to the user database first, we get past the gateway but then cannot reach over to `master` for the login
- if we reach to master directly, we are failing at the gateway with error 40532 state 4 "Cannot open server 'servername' requested by the login"
  So it is possible that the connections succeeds towards gateway or SQL, but not across master <--> boundary.

Let's check the known server names that are used for internal routing:

```
let srv = "servername";
let db  = "databasename";
let startTime = datetime(2023-03-22 02:00:00Z);
let endTime = datetime(2023-03-22 03:00:00Z);
MonLogin
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| filter logical_server_name =~ srv
//| where database_name =~ "master" or database_name =~ db
| where event == "process_login_finish"
| project originalEventTimestamp, NodeName, AppName, LogicalServerName, logical_server_name, server_name, sni_

Sample output:

originalEventTimestamp  NodeName  AppName       LogicalServerName  logical_server_name  server_name
----------------------- --------- ------------- ------------------ -------------------- ----------------------
2023-03-22 02:01:14     GW.42     Gateway       DummyValue         servername           tcp:fo1srv.database.wi
2023-03-22 02:01:14     DB.58     a6c044077814  servername         servername           a6c044077814.tr247.wes
2023-03-22 02:01:14     DB.58     Worker        DummyValue         servername           a6c044077814.tr247.wes
2023-03-22 02:01:14     GW.77     Gateway       DummyValue         servername           tcp:servername.databas
2023-03-22 02:01:17     DB.58     Worker        DummyValue         servername           a6c044077814.tr247.wes
2023-03-22 02:01:17     DB.58     a6c044077814  servername         servername           a6c044077814.tr247.wes
2023-03-22 02:01:27     DB.58     Worker        DummyValue         servername           a6c044077814.tr247.wes
2023-03-22 02:01:27     GW.14     Gateway       DummyValue         servername           tcp:fo2srv.database.wi
2023-03-22 02:01:27     DB.58     a6c044077814  servername         servername           a6c044077814.tr247.wes
2023-03-22 02:01:44     GW.14     Gateway       DummyValue         servername           tcp:servername.databas
2023-03-22 02:01:53     DB.58     Worker        DummyValue         servername           a6c044077814.tr247.wes
2023-03-22 02:01:53     GW.53     Gateway       DummyValue         servername           tcp:fo2srv.database.wi
2023-03-22 02:01:53     DB.58     a6c044077814  servername         servername           a6c044077814.tr247.wes
```

Note how `LogicalServerName` and `logical_server_name` show either a dummy or the correct name, but `server_name` and `sni_server_name` also show different names like "fo1srv.database.windows.net" and "fo2srv.database.windows.net". The server names starting with "a6c044077814" represent the AppName of the server; these are valid internal names for the resource on the tenant ring.

The names "foXsrv.database.windows.net" are indicating the presence of additional DNS names for this server, possibly originating from a Failover Group configuration. Connections to the Failover Group endpoint work, but the server FQDN doesn't and result in errors 40532.

4. Check DMS for sql_alias_cache_records

```
select * from sql_alias_cache_records where (server_name = 'servername' or server_name = 'fo1srv' or server_na
```

In this case, there were no aliases for master and the servername, but for master and the Failover Group.

5. Connectivity checker

If connections are consistently failing, it is a good idea to run the Azure SQL Connectivity Checker. It can provide insights to understand if this is rather a networking issue or an issue with our service.

In this scenario, however, it just confirms the errors that the customer has seen from other clients - but if the configuration would be off, you would be able to see it on the Connectivity Checker output:

```
##########################################################
SUMMARY:
##########################################################
 Found DNS record in DNS server (IP Address:52.236.184.163)
 Found DNS record in Open DNS (IP Address:52.236.184.163)

 Gateway connectivity to 40.68.37.158:1433 FAILED
 Gateway connectivity to 104.40.168.105:1433 succeed
 Gateway connectivity to 52.236.184.163:1433 succeed

 Connection to database master failed (error 40532, state 1): Cannot open server "servername.database.windows.

 Connectivity to login.windows.net:443 succeed (used for AAD Password and Integrated Authentication)
 Connectivity to login.microsoftonline.com:443 succeed (used for AAD Universal with MFA authentication)
 Connectivity to secure.aadcdn.microsoftonline-p.com:443 succeed (used for AAD Universal with MFA authenticati
```

Mitigation

The mitigation in this case is to set an alias for the correct server name and master. You can't do this yourself, but have to open an IcM for it. The PG can run a command for that at the backend to add the alias for resolving the issue.

Internal Reference

- IcM 376637444 - Database xxx on server xxx is not currently available