Error 4014

Last updated by | Holger Linke | Mar 1, 2023 at 4:43 AM PST

Contents

Issue

The customer experiences intermittent, unexpected disconnects of their database connections. The issue appears to affect either larger SQL queries, e.g. with large SQL texts and/or large parameter values, or with large result sets being returned to the client.

The customer might be unable to provide clear error messages or patterns. The common symptoms are dropped connections, either shortly after the login or later when executing queries on existing connections.

When you check the telemetry (Kusto and ASC), you might see a variety of errors:

> **MonSQLSystemHealth (SQL errorlog):**
> Error: 4014, Severity: 20, State: 13.
> A fatal error occurred while reading the input stream from the network. The session will be terminated (input error: 64, output error: 0).

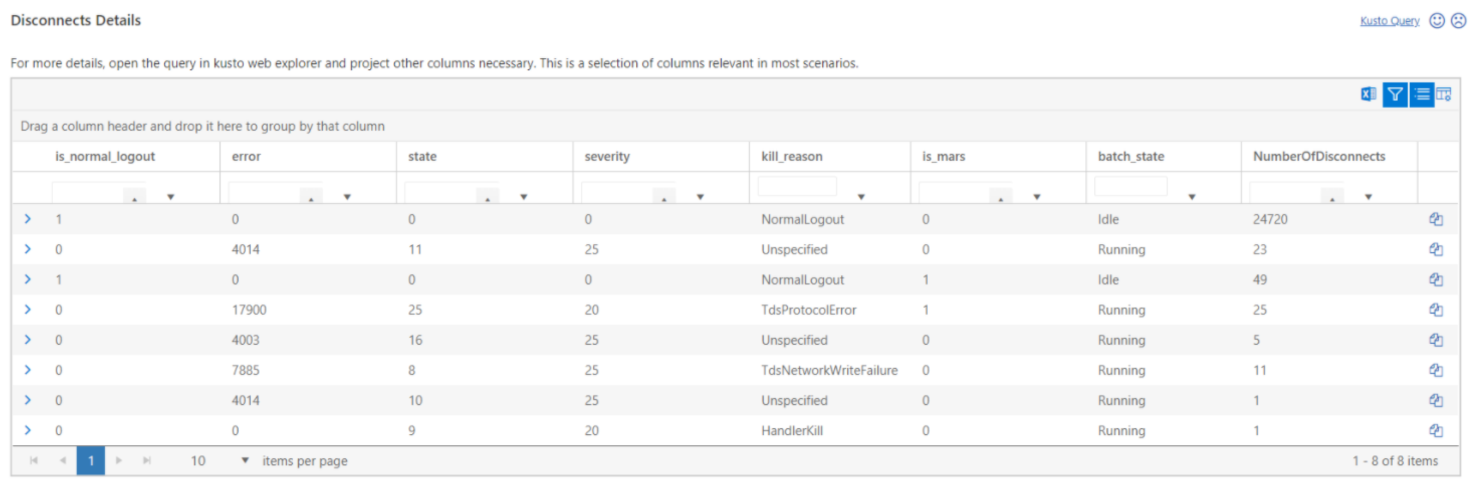
> **MonLogin:**
> error 4014 state 13
> error 17900 state 25
> os_error 50 together with sni_consumer_error 4014 state 13 (for the same connection_id as the error 17900)

> **ASC - Connectivity - Disconnects details:**
> error 4014 with state 10 and 11
> error 7885 state 8
> error 4003 state 16
> error 17900 state 25

Investigation

Using Azure Support Center

Create an ASC troubleshooter report and check the "Connectivity -> Disconnects details" page. The output for this issue will look similar to this:


Disconnects Details

Kusto Query

For more details, open the query in kusto web explorer and project other columns necessary. This is a selection of columns relevant in most scenarios.

Drag a column header and drop it here to group by that column

is_normal_logout	error	state	severity	kill_reason	is_mars	batch_state	NumberOfDisconnects
1	0	0	0	NormalLogout	0	Idle	24720
0	4014	11	25	Unspecified	0	Running	23
1	0	0	0	NormalLogout	1	Idle	49
0	17900	25	20	TdsProtocolError	1	Running	25
0	4003	16	25	Unspecified	0	Running	5
0	7885	8	25	TdsNetworkWriteFailure	0	Running	11
0	4014	10	25	Unspecified	0	Running	1
0	0	9	20	HandlerKill	0	Running	1

1 10 items per page 1 - 8 of 8 items

Using Kusto

Check the SQL errorlog details in `MonSQLSystemHealth` for the time period provided by the customer:

```
let srv = "servername";
let startTime = datetime(2023-02-16 04:00:00Z);
let endTime = datetime(2023-02-16 05:00:00Z);
let timeRange = ago(1d);
MonSQLSystemHealth
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| project originalEventTimestamp, NodeName, AppName, error_id, message
| order by originalEventTimestamp asc
| limit 1000
```

> Result:
> Error: 4014, Severity: 20, State: 13.
> A fatal error occurred while reading the input stream from the network. The session will be terminated (input error: 64, output error: 0).

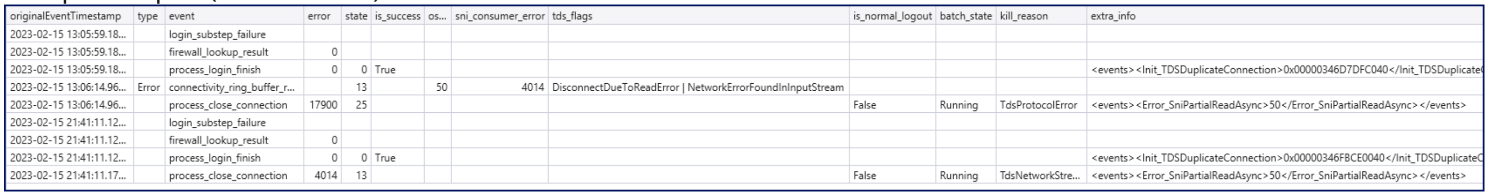
In `MonLogin` , you may also see errors 4014 and 17900 returned:

```
// First - narrow down on the errors:
let srv = "servername";
let startTime = datetime(2023-02-16 04:00:00Z);
let endTime = datetime(2023-02-16 05:00:00Z);
let timeRange = ago(1d);
MonLogin
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where event == "process_close_connection" and kill_reason <> 'NormalLogout'
| where logical_server_name =~ srv
| project PreciseTimeStamp, ClusterName, NodeName, AppName, event, logical_server_name, database_name, error,
| sort by PreciseTimeStamp desc

Sample output (abbreviated):
PreciseTimeStamp     event                     error  state  connection_id                         kill_reason
-------------------- ------------------------- ------ ------ ------------------------------------- -----------
2023-02-16 04:59:23  process_close_connection  17900  25     C8654FFA-05CE-439A-9388-1CBF14F5FE9A  TdsProtocol
2023-02-16 04:58:53  process_close_connection  4014   13     75137019-B8D5-4B8D-A3B5-CDFB1645DD33  TdsNetworkS
2023-02-16 04:58:53  process_close_connection  17900  25     9683C5CE-6ECE-4FFC-A9C7-9C299D3EA1DF  TdsProtocol
2023-02-16 04:58:53  process_close_connection  17900  25     E8968148-5E24-42F8-851C-BE2166603F29  TdsProtocol


// Second - check details for some of the connection_ids
let srv = "servername";
let db = "databasename";
let startTime = datetime(2023-02-16 04:00:00Z);
let endTime = datetime(2023-02-16 05:00:00Z);
let timeRange = ago(1d);
MonLogin
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where LogicalServerName contains srv or logical_server_name contains srv or server_name contains srv //or in
//| where database_name =~ db
| where event !has "db_fw_cache"
| where connection_id in ("connection_id_1", "connection_id_2")
| extend ProxyOrRedirect = iif( result =~ "e_crContinue", "Redirect", iif( result =~ "e_crContinueSameState",
| extend fedauth_library_type_desc =
    case(
        fedauth_library_type == 0, "SQL Server Authentication",
        fedauth_library_type == 2, "Token Base Authentication",
        fedauth_library_type == 3 and fedauth_adal_workflow == 1, "Azure Active Directory - Password Authentic
        fedauth_library_type == 3 and fedauth_adal_workflow == 2, "Azure Active Directory - Integrated Authent
        fedauth_library_type == 3 and fedauth_adal_workflow == 3, "Azure Active Directory - Universal Authenti
        strcat(tostring(fedauth_library_type) , "-" , tostring(fedauth_adal_workflow))
    )
| extend AADUser = iif( fedauth_adal_workflow > 0 or fedauth_library_type > 0, "AAD" , "SQL")
| project originalEventTimestamp, type, event, error, state, is_user_error, is_success, os_error, sni_error, s
| limit 1000
| order by originalEventTimestamp asc
```

Sample output (abbreviated):

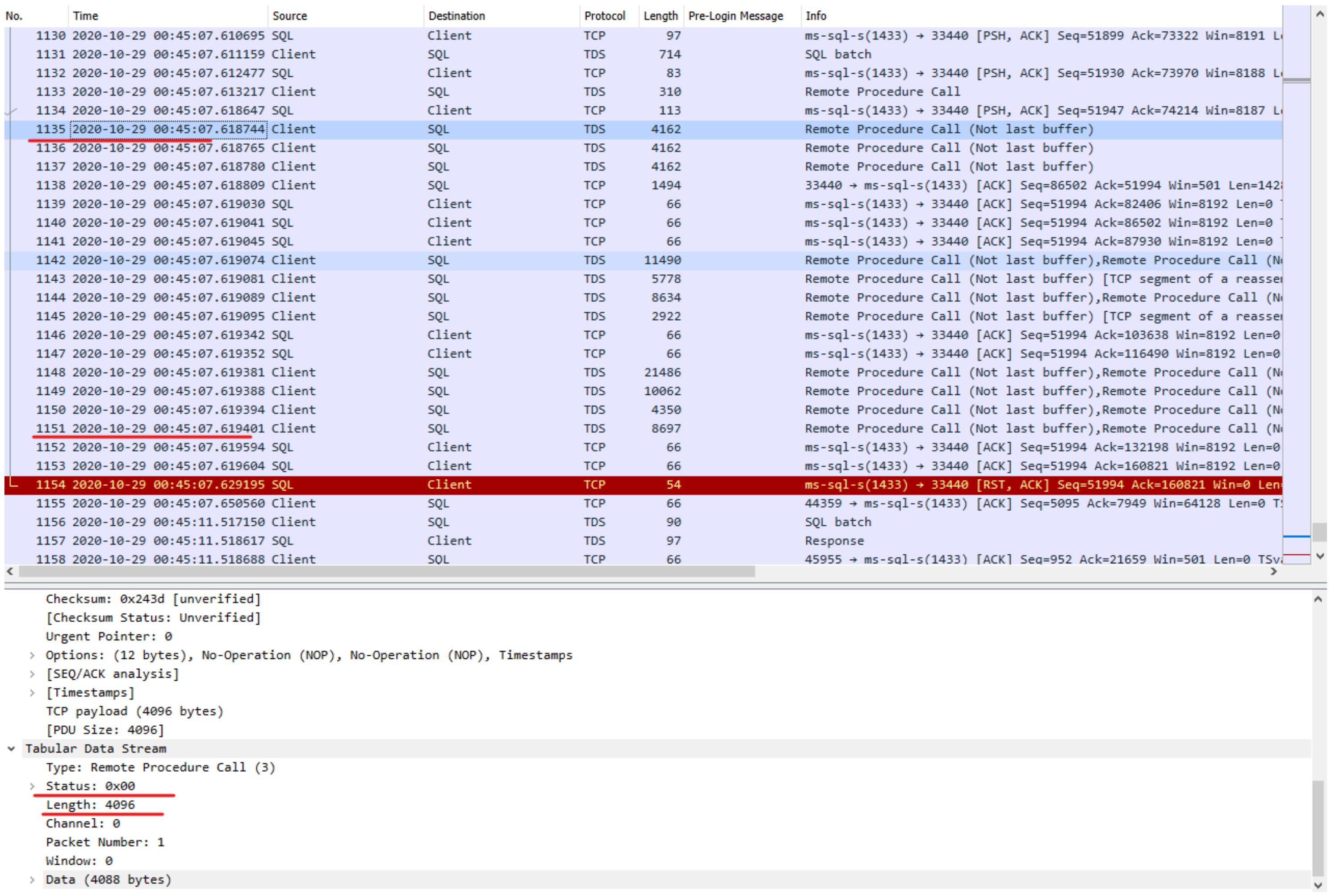


originalEventTimestamp	type	event	error	state	is_success	os...	sni_consumer_error	tds_flags	is_normal_logout	batch_state	kill_reason	extra_info
2023-02-15 13:05:59.18...		login_substep_failure										
2023-02-15 13:05:59.18...		firewall_lookup_result	0									
2023-02-15 13:05:59.18...		process_login_finish	0	0	True							<events><Init_TDSDuplicateConnection>0x00000346D7DFC040</Init_TDSDuplicate
2023-02-15 13:06:14.96...	Error	connectivity_ring_buffer_r...		13		50	4014	DisconnectDueToReadError | NetworkErrorFoundInInputStream				
2023-02-15 13:06:14.96...		process_close_connection	17900	25					False	Running	TdsProtocolError	<events><Error_SniPartialReadAsync>50</Error_SniPartialReadAsync></events>
2023-02-15 21:41:11.12...		login_substep_failure										
2023-02-15 21:41:11.12...		firewall_lookup_result	0									
2023-02-15 21:41:11.12...		process_login_finish	0	0	True							<events><Init_TDSDuplicateConnection>0x00000346FBCE0040</Init_TDSDuplicate
2023-02-15 21:41:11.17...		process_close_connection	4014	13					False	Running	TdsNetworkStre...	<events><Error_SniPartialReadAsync>50</Error_SniPartialReadAsync></events>

To investigate any further, we need to capture a network trace at client side.

Using Network Trace

From the packet trace, we can see after a series of RPC request the server is resetting the connection:



```
No.  Time                       Source  Destination  Protocol  Length  Pre-Login Message  Info
1130 2020-10-29 00:45:07.610695 SQL     Client       TCP          97                      ms-sql-s(1433) → 33440 [PSH, ACK] Seq=51899 Ack=73322 Win=8191 L
1131 2020-10-29 00:45:07.611159 Client  SQL          TDS         714                      SQL batch
1132 2020-10-29 00:45:07.612477 SQL     Client       TCP          83                      ms-sql-s(1433) → 33440 [PSH, ACK] Seq=51930 Ack=73970 Win=8188 L
1133 2020-10-29 00:45:07.613217 Client  SQL          TDS         310                      Remote Procedure Call
1134 2020-10-29 00:45:07.618647 SQL     Client       TCP         113                      ms-sql-s(1433) → 33440 [PSH, ACK] Seq=51947 Ack=74214 Win=8187 L
1135 2020-10-29 00:45:07.618744 Client  SQL          TDS        4162                      Remote Procedure Call (Not last buffer)
1136 2020-10-29 00:45:07.618765 Client  SQL          TDS        4162                      Remote Procedure Call (Not last buffer)
1137 2020-10-29 00:45:07.618780 Client  SQL          TDS        4162                      Remote Procedure Call (Not last buffer)
1138 2020-10-29 00:45:07.618809 Client  SQL          TCP        1494                      33440 → ms-sql-s(1433) [ACK] Seq=86502 Ack=51994 Win=501 Len=142
1139 2020-10-29 00:45:07.619030 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=82406 Win=8192 Len=0
1140 2020-10-29 00:45:07.619041 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=86502 Win=8192 Len=0
1141 2020-10-29 00:45:07.619045 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=87930 Win=8192 Len=0
1142 2020-10-29 00:45:07.619074 Client  SQL          TDS       11490                      Remote Procedure Call (Not last buffer),Remote Procedure Call (N
1143 2020-10-29 00:45:07.619081 Client  SQL          TDS        5778                      Remote Procedure Call (Not last buffer) [TCP segment of a reasse
1144 2020-10-29 00:45:07.619089 Client  SQL          TDS        8634                      Remote Procedure Call (Not last buffer),Remote Procedure Call (N
1145 2020-10-29 00:45:07.619095 Client  SQL          TDS        2922                      Remote Procedure Call (Not last buffer) [TCP segment of a reasse
1146 2020-10-29 00:45:07.619342 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=103638 Win=8192 Len=0
1147 2020-10-29 00:45:07.619352 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=116490 Win=8192 Len=0
1148 2020-10-29 00:45:07.619381 Client  SQL          TDS       21486                      Remote Procedure Call (Not last buffer),Remote Procedure Call (N
1149 2020-10-29 00:45:07.619388 Client  SQL          TDS       10062                      Remote Procedure Call (Not last buffer),Remote Procedure Call (N
1150 2020-10-29 00:45:07.619394 Client  SQL          TDS        4350                      Remote Procedure Call (Not last buffer),Remote Procedure Call (N
1151 2020-10-29 00:45:07.619401 Client  SQL          TDS        8697                      Remote Procedure Call (Not last buffer),Remote Procedure Call (N
1152 2020-10-29 00:45:07.619594 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=132198 Win=8192 Len=0
1153 2020-10-29 00:45:07.619604 SQL     Client       TCP          66                      ms-sql-s(1433) → 33440 [ACK] Seq=51994 Ack=160821 Win=8192 Len=0
1154 2020-10-29 00:45:07.629195 SQL     Client       TCP          54                      ms-sql-s(1433) → 33440 [RST, ACK] Seq=51994 Ack=160821 Win=0 Len
1155 2020-10-29 00:45:07.650560 Client  SQL          TCP          66                      44359 → ms-sql-s(1433) [ACK] Seq=5095 Ack=7949 Win=64128 Len=0 T
1156 2020-10-29 00:45:11.517150 Client  SQL          TDS          90                      SQL batch
1157 2020-10-29 00:45:11.518617 SQL     Client       TDS          97                      Response
1158 2020-10-29 00:45:11.518688 Client  SQL          TCP          66                      45955 → ms-sql-s(1433) [ACK] Seq=952 Ack=21659 Win=501 Len=0 TSv

    Checksum: 0x243d [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (4096 bytes)
    [PDU Size: 4096]
v Tabular Data Stream
    Type: Remote Procedure Call (3)
  > Status: 0x00
    Length: 4096
    Channel: 0
    Packet Number: 1
    Window: 0
  > Data (4088 bytes)
```

Status : 0x00 indicates that this is not the end of the message (EOM).
So the client is sending a big request in multiple 4K packets starting at packet 1135 which terminates in packet 1151 with Status: 0x01.

This is a common scenario that big request are cut into small packets for network transfer. This may cause issues if some of the packets are lost, usually due to a problem on the network. Involve the Azure Networking support team if you need further analysis and details.

Analysis

Error 4014 indicates that SQL has started reading a message from the client which potentially spans multiple TDS packets. State 13 means that a read failed after we attempted to read the remainder of a partially-read packet. See the "More Information" section below for the other errors.

The results from the Investigation appear to contradict each other: The network trace shows that the server resets the connection, pointing to the Azure gateway or SQL Database. The gateway telemetry however shows that the issue is on the client-side, outside of Azure. The cause of the issue though usually is either on the client network side or with the client-side SQL driver/provider and needs to be analyzed further from the application and client side.

The issue is known to occur if the client application is using the jTDS JDBC driver. jTDS is an open source Java (type 4) JDBC 3.0 driver for Microsoft SQL Server 6.5, 7, 2000, 2005, 2008 and 2012. The latest version is from June 2013; it has no concept of cloud databases, proxy vs. redirect, retry logic, network packet handling over intermediate proxies and gateways, or any feature that had been introduced after SQL Server 2012. jTDS is typical for an on-premise application that has been moved unchanged into Azure and now has to work in an environment it hadn't been designed for.

Mitigation

Increase TDS packet size

Configure a larger TDS packet size to minimize the impact. This might not fully resolve the issue but decreases the likelihood for the issue to occur.

Examples for setting the packet size:

- If the client app is a tool like bcp, set the "-a" parameter to a large size.
- For JDBC clients, the packet size can be configured in the packetSize connection property.

Update client-side drivers

Check the version of the client-side SQL Server drivers and update them to the newest version. This depends on the application; it could be an ODBC driver, OLE DB provider, JDBC driver, SQL Native Client driver; platforms could be either Windows, Linux or even Android.

jTDS driver

The jTDS driver is not cloud-aware and needs to be replaced with a current JDBC driver. Depending on the application, this could be a major software development project for the customer; it might be an easy step in a home-grown application or rather impossible if it's a legacy third-party application.

There is a very high probability that there is no workaround for the issue, and replacing the jTDS driver is the only valid option for mitigation.

More Information

Common error codes related to this issue:

Error	Definition
4003	This is a control exception that is used to transfer control during request processing, after the real exception has already been logged.
4014	A fatal error occurred while reading the input stream from the network. The session will be terminated (input error: %d, output error: %d).
7885	Network error 0x%lx occurred while sending data to the client on process ID %d batch ID %d. A common cause for this error is if the client disconnected without reading the entire response from the server. This connection will be terminated.
17900	A network error occurred in the established connection; the connection has been closed.

Classification

Root Cause: Azure SQL DB v3\Connectivity\Disconnects\Other

How good have you found this content?