Troubleshooting PostgreSQL Replica issues

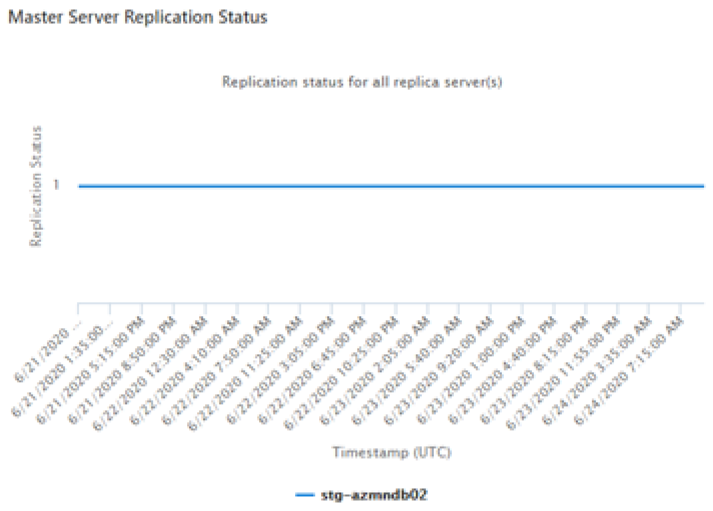
Last updated by | Lisa Liu | Nov 6, 2020 at 10:35 AM PST

Check if the replication status/Latency from ASC:

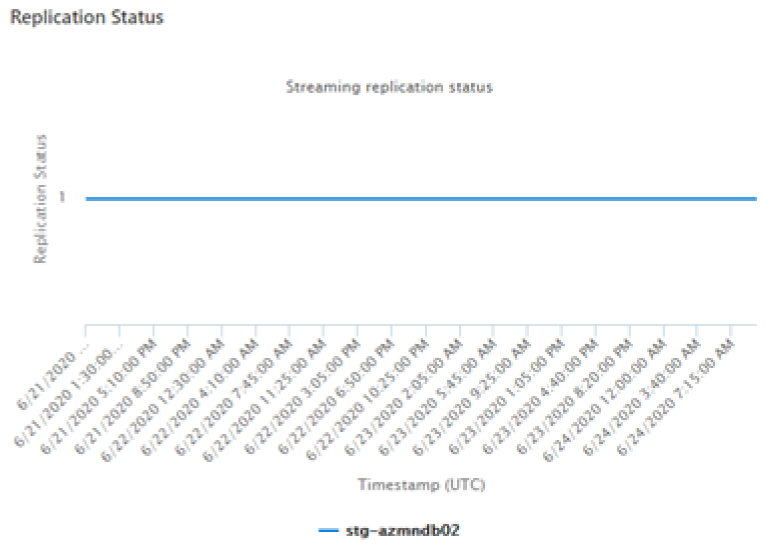
Replication status:

You can check the replication status/Lag from the Master or from the Slave in the ASC replication tab, value of 1 means the replication is running , 0 if it is not.

From Primary:

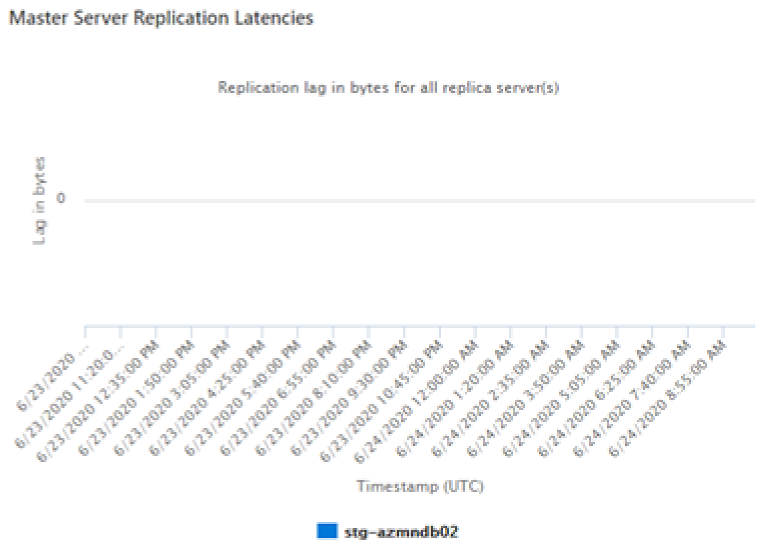


From Slave:

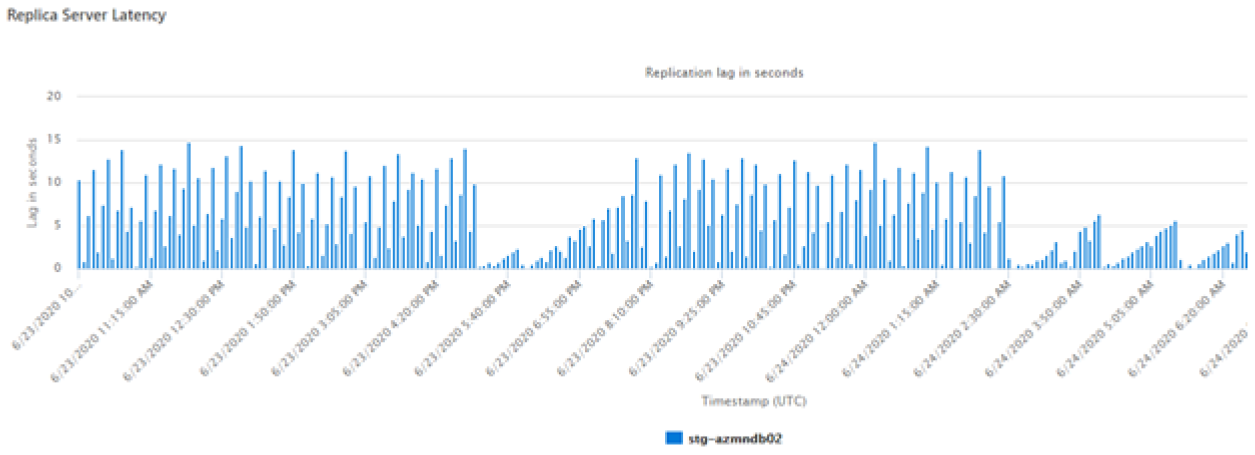


Replication Lag:

From Primary:



From Slave:

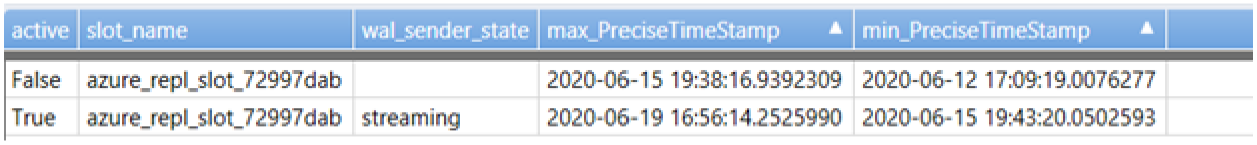


You can see there is a latency here , and this coming from replaying the transaction on the slave , and no problem on the master as we noticed because we can see a 0 lag in the master database, on this scenario check the slave resources , CPU,and IOPs to check if there is any contention, scaling the storage in the slave in our case minimized the latency

<u>Check if the replication status/Latency from Kusto:</u>

Replication status:

```
MonDmPgSqlReplicationStatsPrimary
| where TIMESTAMP > ago (7d)
| where LogicalServerName == 'jirasd-prod-db2'
| project PreciseTimeStamp, LogicalServerName, AppName, slot_name, active, wal_sender_state,
application_name, total_lag_in_bytes
|summarize max(PreciseTimeStamp) , min(PreciseTimeStamp) by active ,  slot_name, wal_sender_state
```



active	slot_name	wal_sender_state	max_PreciseTimeStamp	min_PreciseTimeStamp
False	azure_repl_slot_72997dab		2020-06-15 19:38:16.9392309	2020-06-12 17:09:19.0076277
True	azure_repl_slot_72997dab	streaming	2020-06-19 16:56:14.2525990	2020-06-15 19:43:20.0502593

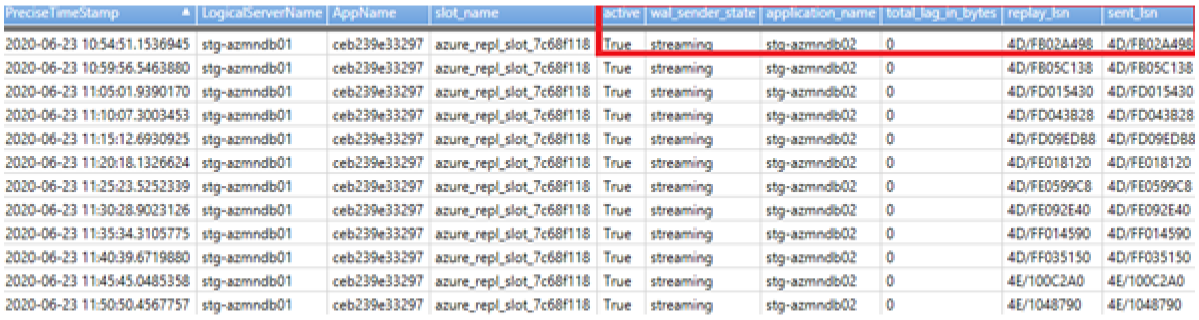
If you find multiple rows here , so the replication stopped/was stopped , depends on the timestamp , for example from the above query we can see that the replication was broken between 2020-06-12 17:19:25.3924527 and 2020-06-15 19:38:16.9392309 and now it is working .

Replication Latency:

From Primary

```
MonDmPgSqlReplicationStatsPrimary
| where TIMESTAMP > ago (1d)
| where LogicalServerName == 'stg-azmndb01'
| project PreciseTimeStamp, LogicalServerName, AppName, slot_name, active, wal_sender_state,
application_name, total_lag_in_bytes,replay_lsn,sent_lsn
```

-



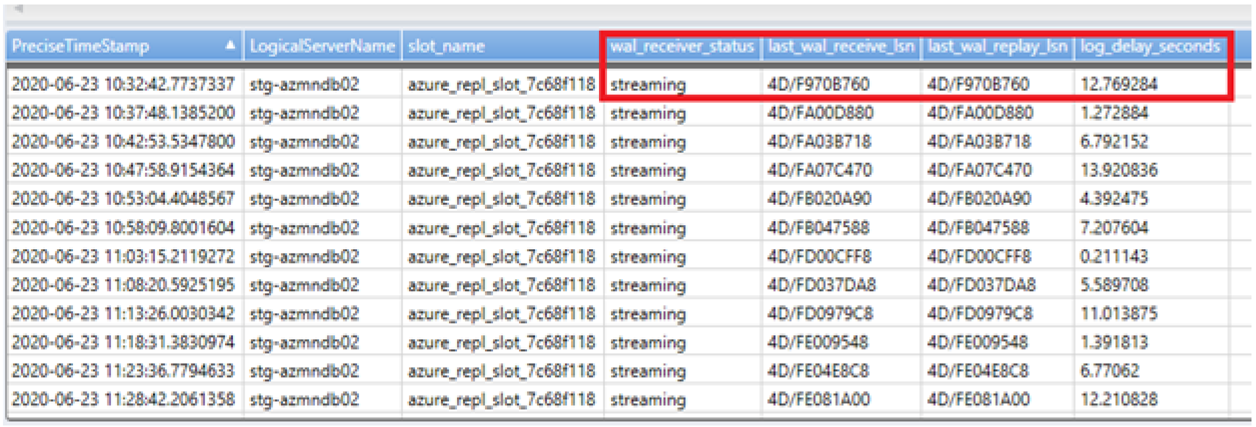
PreciseTimeStamp	LogicalServerName	AppName	slot_name	active	wal_sender_state	application_name	total_lag_in_bytes	replay_lsn	sent_lsn
2020-06-23 10:54:51.1536945	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FB02A498	4D/FB02A498
2020-06-23 10:59:56.5463880	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FB05C138	4D/FB05C138
2020-06-23 11:05:01.9390170	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FD015430	4D/FD015430
2020-06-23 11:10:07.3003453	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FD043828	4D/FD043828
2020-06-23 11:15:12.6930925	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FD09EDB8	4D/FD09EDB8
2020-06-23 11:20:18.1326624	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FE018120	4D/FE018120
2020-06-23 11:25:23.5252339	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FE0599C8	4D/FE0599C8
2020-06-23 11:30:28.9023126	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FE092E40	4D/FE092E40
2020-06-23 11:35:34.3105775	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FF014590	4D/FF014590
2020-06-23 11:40:39.6719880	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4D/FF035150	4D/FF035150
2020-06-23 11:45:45.0485358	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4E/100C2A0	4E/100C2A0
2020-06-23 11:50:50.4567757	stg-azmndb01	ceb239e33297	azure_repl_slot_7c68f118	True	streaming	stg-azmndb02	0	4E/1048790	4E/1048790

The replay_lsn on the replica is the same as the sent_lsn on the master, which is an indication there is no lag here and the total_lag_in_bytes is 0 too.

From Slave:

```
MonDmPgSqlReplicationStatsPrimary
| where TIMESTAMP > ago (1d)
| where LogicalServerName == 'stg-azmndb01'

  | project PreciseTimeStamp, LogicalServerName, last_wal_receive_lsn,last_wal_replay_lsn,log_delay_seconds
```



PreciseTimeStamp	LogicalServerName	slot_name	wal_receiver_status	last_wal_receive_lsn	last_wal_replay_lsn	log_delay_seconds
2020-06-23 10:32:42.7737337	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/F970B760	4D/F970B760	12.769284
2020-06-23 10:37:48.1385200	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FA00D880	4D/FA00D880	1.272884
2020-06-23 10:42:53.5347800	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FA03B718	4D/FA03B718	6.792152
2020-06-23 10:47:58.9154364	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FA07C470	4D/FA07C470	13.920836
2020-06-23 10:53:04.4048567	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FB020A90	4D/FB020A90	4.392475
2020-06-23 10:58:09.8001604	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FB047588	4D/FB047588	7.207604
2020-06-23 11:03:15.2119272	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FD00CFF8	4D/FD00CFF8	0.211143
2020-06-23 11:08:20.5925195	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FD037DA8	4D/FD037DA8	5.589708
2020-06-23 11:13:26.0030342	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FD0979C8	4D/FD0979C8	11.013875
2020-06-23 11:18:31.3830974	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FE009548	4D/FE009548	1.391813
2020-06-23 11:23:36.7794633	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FE04E8C8	4D/FE04E8C8	6.77062
2020-06-23 11:28:42.2061358	stg-azmndb02	azure_repl_slot_7c68f118	streaming	4D/FE081A00	4D/FE081A00	12.210828

As you can see there, last_wal_receive_lsn is the same as last_wal_replay_lsn and the time taken to replay here is about 12 seconds , and note that this is not necessary a problem as in some cases it is expected.

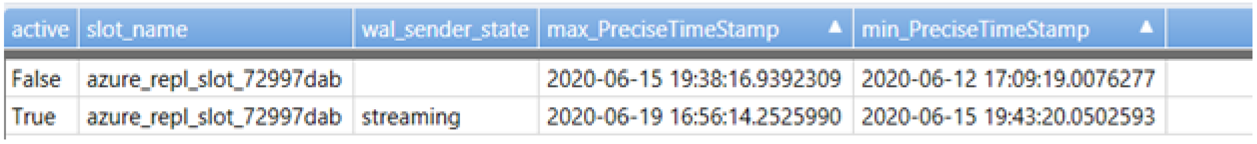
-

Note:

if you want to check why the replication was/is broken, you can follow the below steps:

Check the timestamp when the replication was broken

```
MonDmPgSqlReplicationStatsPrimary
| where TIMESTAMP > ago (7d)
| where LogicalServerName == 'jirasd-prod-db2'
| project PreciseTimeStamp, LogicalServerName, AppName, slot_name, active, wal_sender_state,
application_name, total_lag_in_bytes
|summarize max(PreciseTimeStamp) , min(PreciseTimeStamp) by active ,  slot_name, wal_sender_state
```

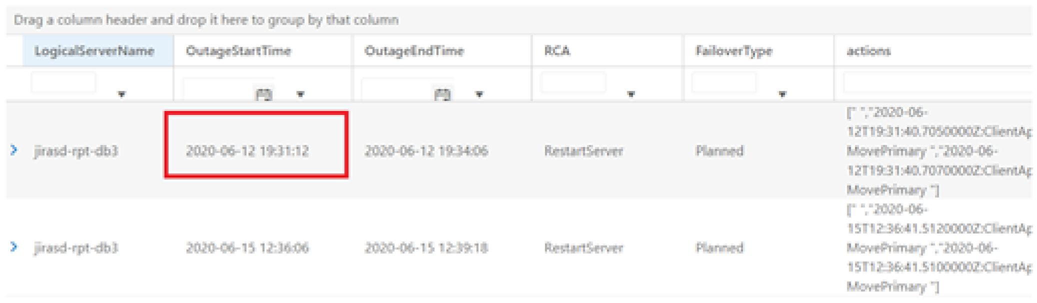


active	slot_name	wal_sender_state	max_PreciseTimeStamp	min_PreciseTimeStamp
False	azure_repl_slot_72997dab		2020-06-15 19:38:16.9392309	2020-06-12 17:09:19.0076277
True	azure_repl_slot_72997dab	streaming	2020-06-19 16:56:14.2525990	2020-06-15 19:43:20.0502593

we can see that the replication was broken between 2020-06-12 17:19:25.3924527 and 2020-06-15 19:38:16.9392309 and now it is working.

Check if there is any restart happened on replica/master, so after the restart it will take some time to restart the streaming replication:

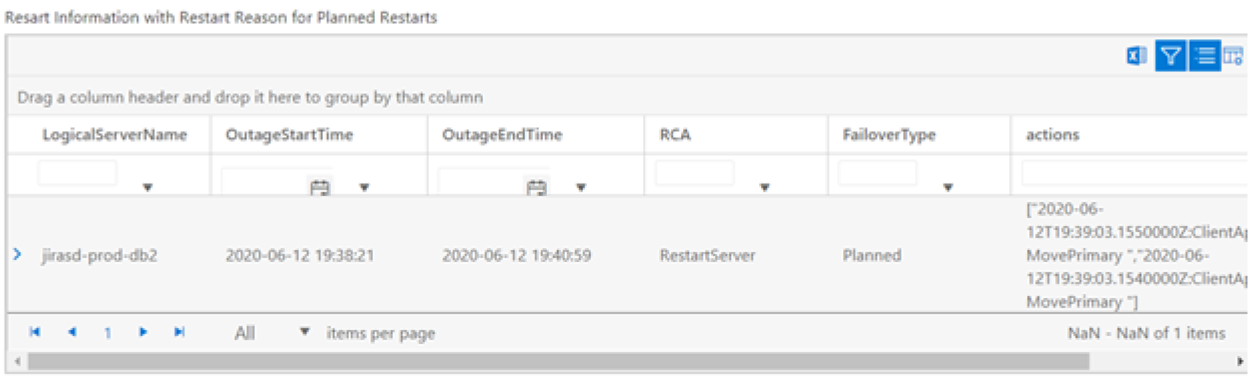
Based on the above timestamp results, we can see that the customer initiated a replica restart:



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

	LogicalServerName	OutageStartTime	OutageEndTime	RCA	FailoverType	actions
›	jirasd-rpt-db3	2020-06-12 19:31:12	2020-06-12 19:34:06	RestartServer	Planned	[" ","2020-06-12T19:31:40.7050000Z:ClientAp MovePrimary ","2020-06-12T19:31:40.7070000Z:ClientAp MovePrimary "]
›	jirasd-rpt-db3	2020-06-15 12:36:06	2020-06-15 12:39:18	RestartServer	Planned	[" ","2020-06-15T12:36:41.5120000Z:ClientAp MovePrimary ","2020-06-15T12:36:41.5100000Z:ClientAp MovePrimary "]

And also, from the primary side, but both are after the replication broken:



Resart Information with Restart Reason for Planned Restarts

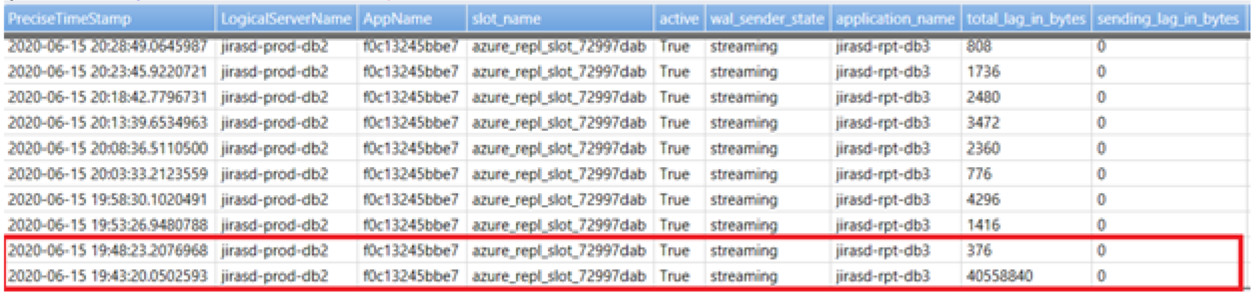
Drag a column header and drop it here to group by that column

	LogicalServerName	OutageStartTime	OutageEndTime	RCA	FailoverType	actions
›	jirasd-prod-db2	2020-06-12 19:38:21	2020-06-12 19:40:59	RestartServer	Planned	["2020-06-12T19:39:03.1550000Z:ClientAp MovePrimary ","2020-06-12T19:39:03.1540000Z:ClientAp MovePrimary "]

1 All items per page NaN - NaN of 1 items

And after 2020-06-15 19:38:16.9392309 the replication starts to work:

```
MonDmPgSqlReplicationStatsPrimary
| where TIMESTAMP > ago (7d)
| where LogicalServerName == 'jirasd-prod-db2'
| where active =='True'
| project PreciseTimeStamp, LogicalServerName, AppName, slot_name, active, wal_sender_state,
application_name, total_lag_in_bytes
| order by PreciseTimeStamp desc
```



PreciseTimeStamp	LogicalServerName	AppName	slot_name	active	wal_sender_state	application_name	total_lag_in_bytes	sending_lag_in_bytes
2020-06-15 20:28:49.0645987	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	808	0
2020-06-15 20:23:45.9220721	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	1736	0
2020-06-15 20:18:42.7796731	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	2480	0
2020-06-15 20:13:39.6534963	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	3472	0
2020-06-15 20:08:36.5110500	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	2360	0
2020-06-15 20:03:33.2123559	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	776	0
2020-06-15 19:58:30.1020491	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	4296	0
2020-06-15 19:53:26.9480788	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	1416	0
2020-06-15 19:48:23.2076968	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	376	0
2020-06-15 19:43:20.0502593	jirasd-prod-db2	f0c13245bbe7	azure_repl_slot_72997dab	True	streaming	jirasd-rpt-db3	40558840	0

Let's see the sandbox to check if there are any errors, and the timestamp will be around the time when the replica stopped working:

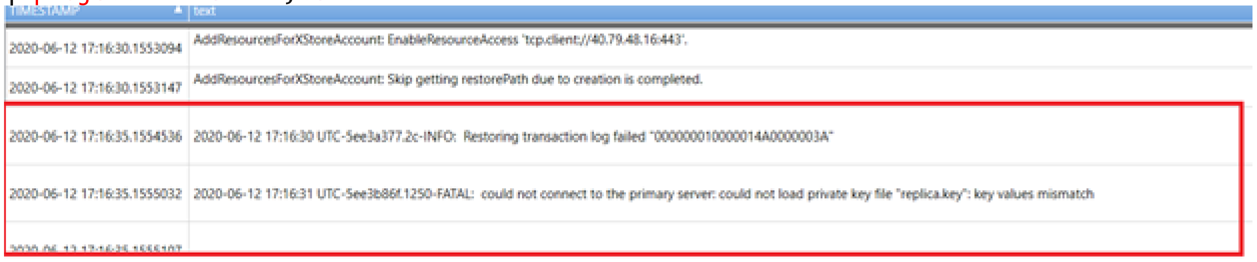
On primary:

```
MonRdmsPgSqlSandbox
| where LogicalServerName == 'jirasd-rpt-db3'
| where TIMESTAMP >= datetime(2020-06-12 17:16:19.0076277 ) and TIMESTAMP <= datetime(2020-06-12
17:21:19.0076277 )
| project TIMESTAMP,text
```

No suspicious messages, all is good.

```
On Replica:

MonRdmsPgSqlSandbox
| where LogicalServerName == 'jirasd-rpt-db3'
| where TIMESTAMP >= datetime(2020-06-12 17:16:19.0076277 ) and TIMESTAMP <= datetime(2020-06-12
17:21:19.0076277 )
| project TIMESTAMP,text
```



TIMESTAMP	text
2020-06-12 17:16:30.1553094	AddResourcesForXStoreAccount: EnableResourceAccess 'tcp.client://40.79.48.16:443'.
2020-06-12 17:16:30.1553147	AddResourcesForXStoreAccount: Skip getting restorePath due to creation is completed.
2020-06-12 17:16:35.1554536	2020-06-12 17:16:30 UTC-5ee3a377.2c-INFO: Restoring transaction log failed "000000010000014A0000003A"
2020-06-12 17:16:35.1555032	2020-06-12 17:16:31 UTC-5ee3b86f.1250-FATAL: could not connect to the primary server: could not load private key file "replica.key": key values mismatch
2020-06-12 17:16:35.1555107	

And these error messages are still there until it is resolved on 2020-06-15 19:38:16.9392309 , there was an issue on this server certificate, this is a sample and you may face different messages there.

How good have you found this content?