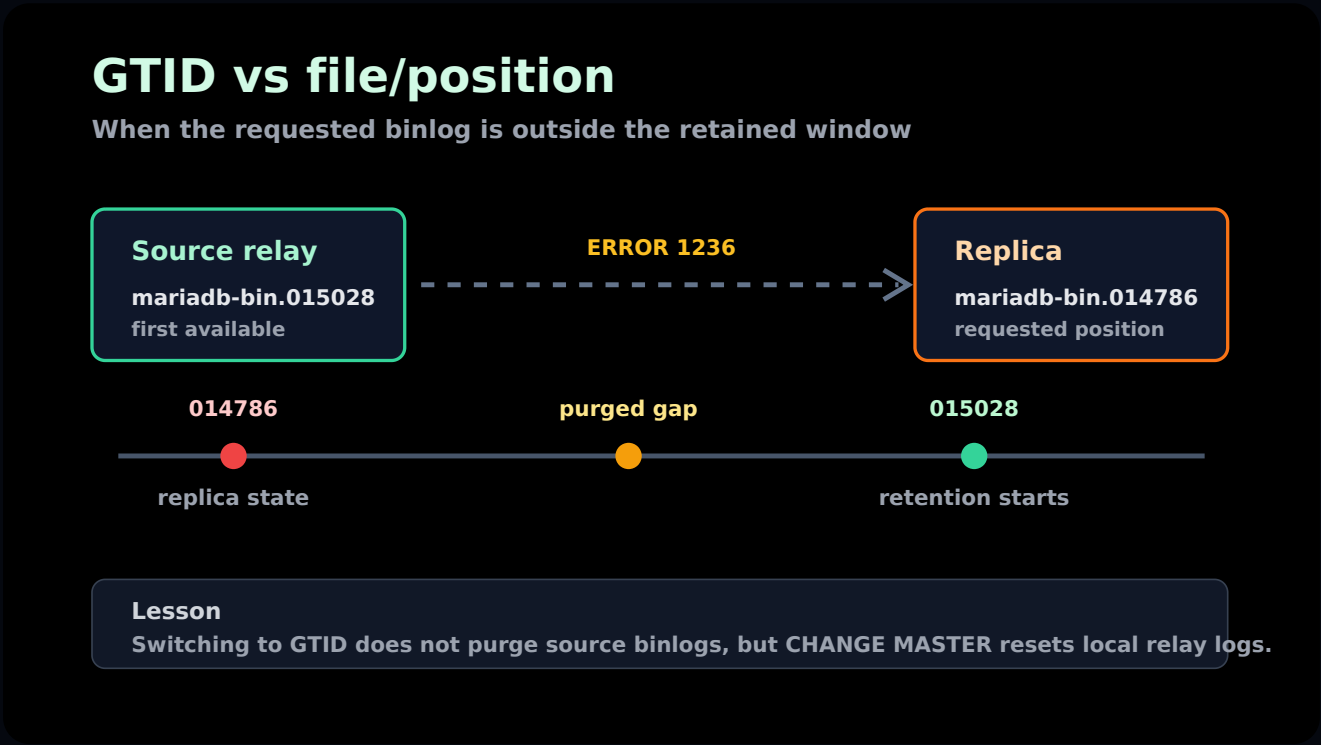
When Switching to GTID Breaks Replication

Aurélien LEQUOY · May 18, 2026

MARIADB MYSQL REPLICATION GTID BINLOG PMACONTROL DBA INCIDENT



The Symptom

The scenario is familiar: MariaDB replication runs in file/position mode, GTID is enabled to test or harden the attachment, then the setup is switched back to normal mode. A few seconds later, the IO thread refuses to start:

```
SHOW SLAVE STATUS\G
```

```
Slave_IO_Running: No
Slave_SQL_Running: Yes
Using_Gtid: No
Last_IO_Errno: 1236
Last_IO_Error: Could not find first log file name in binary log index file
```

The immediate assumption is tempting: "switching to GTID purged the relay logs or the binlogs." In the incident analyzed here, that was not the case.

What the GTID Button Actually Does

In PmaControl, the `GTID > Activate` action on MariaDB replication does not run a purge. It does not run `RESET SLAVE` either.

It only does this:

```
STOP SLAVE;
CHANGE MASTER TO MASTER_USE_GTID = slave_pos;
START SLAVE;
```

The rollback action does this:

```
STOP SLAVE;
CHANGE MASTER TO MASTER_USE_GTID = no;
START SLAVE;
```

These commands do not delete the source's binlogs. However, `CHANGE MASTER` recreates the local replication state and resets the replica's relay logs. That distinction matters: the source loses nothing, but the replica discards what it had already downloaded locally.

Full-Scale Test

To verify the behavior, I reproduced the case on two MariaDB 11.8 lab servers:

```
source -> replica
file/position at the start
relay_log_purge=ON
```

The test:

1. start clean file/position replication;
2. stop only the replica SQL thread;
3. generate 20,000 rows on the source;
4. verify that the IO thread downloads the events into relay logs;

5. execute the same sequence as the `GTID > Activate` button.

Before `CHANGE MASTER`, the replica had unapplied relay logs:

```
Slave_IO_Running: Yes
Slave_SQL_Running: No
Read_Master_Log_Pos: 635180
Exec_Master_Log_Pos: 3464
Relay_Log_Space: 632576
relay-bin.000002: 632273 bytes
```

After `STOP SLAVE`, nothing was lost:

```
Slave_IO_Running: No
Slave_SQL_Running: No
Relay_Log_Space: 632576
relay-bin.000002: 632273 bytes
```

Immediately after:

```
CHANGE MASTER TO MASTER_USE_GTID=slave_pos;
```

the relay logs were reset:

```
Using_Gtid: Slave_Pos
Read_Master_Log_Pos: 3464
Exec_Master_Log_Pos: 3464
Relay_Log_Space: 256
relay-bin.000001: 256 bytes
```

The reverse test gives the same result. In GTID mode, with unapplied relay logs, returning to file/position via:

```
CHANGE MASTER TO MASTER_USE_GTID=no;
```

also resets the local relay logs.

Test conclusion: `STOP SLAVE` alone keeps the relay logs. `CHANGE MASTER`, even when limited to `MASTER_USE_GTID`, resets them.

What Really Happened

The replica returned to file/position mode with a persisted coordinate that was too old:

```
mariadb-bin.014786:35578691
```

The relay source no longer had that file in its binary log index. Its first available entry was already newer:

```
first available: mariadb-bin.015028
last available:  mariadb-bin.015130
retained files:  103
retained size:   about 10 GiB
```

The source was still configured with:

```
binlog_expire_logs_seconds = 864000
expire_logs_days           = 10
binlog_space_limit         = 0
max_binlog_size            = 104857600
```

So the configured retention said "10 days". But the coordinate requested by the replica was older than the window actually available.

Why 10 Days Do Not Guarantee That File

A 10-day retention means the server may purge binary logs older than that limit. It does not guarantee that a replica can return to any old physical coordinate after a mode switch.

Several situations make this dangerous:

- replication was already lagging before the click;
- the replica kept an old file/position coordinate while GTID was enabled;
- a previous purge had already removed the requested files;
- the retention value may have changed in the past;
- maintenance or a rebuild may have recreated a shorter window than expected;
- the upstream master may use a different policy than the relay source.

The key point is simple: MariaDB cannot serve a file that is absent from `SHOW BINARY LOGS`, even if the current configuration displays 10 days.

Why GTID Hid the Problem

In MariaDB GTID mode, the replica no longer asks for a `MASTER_LOG_FILE` / `MASTER_LOG_POS` pair. It asks for a logical set of transactions through `gtid_slave_pos`.

When you switch back to file/position, old physical coordinates become relevant again. If those coordinates point to a purged file, the IO thread immediately fails with `1236`.

GTID did not purge the source binlogs. However, the switch through `CHANGE MASTER` did delete the local relay logs already downloaded by the replica. If the transactions present in those relay logs are no longer available in the source binlogs, recovery becomes impossible without resynchronization.

Why a Simple CHANGE MASTER Is Not Enough

It is tempting to repoint the replica to the current source position:

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_LOG_FILE='mariadb-bin.015130',
  MASTER_LOG_POS=97211185,
  MASTER_USE_GTID=no;
START SLAVE;
```

The IO thread may restart. But the SQL thread may then fail on a constraint, for example:

```
Last_SQL_Errno: 1452
Cannot add or update a child row:
foreign key constraint fails
```

That is expected: jumping directly to a recent position skips a range of transactions. The replica's data no longer matches the sequence of events it receives next.

That is not a repair. It is a jump in the log.

The Correct Fix

The safe fix is a full replica resynchronization from a coherent source:

1. stop replication;
2. take a consistent backup from the source or relay source;
3. restore the replica;
4. retrieve the exact backup position;
5. configure replication in GTID or file/position mode;
6. start replication;
7. verify both threads are healthy.

Expected result:

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Seconds_Behind_Master: 0 or decreasing
Last_IO_Error:
Last_SQL_Error:
```

Avoid `sql_slave_skip_counter` here. Skipping FK errors hides the divergence and leaves inconsistent data behind.

The Check to Add Before Clicking

Before enabling or disabling GTID, PmaControl should be able to answer three questions:

```
SHOW SLAVE STATUS\G
SHOW BINARY LOGS;
SHOW VARIABLES WHERE Variable_name IN (
  'expire_logs_days',
  'binlog_expire_logs_seconds',
  'binlog_space_limit',
  'max_binlog_size'
);
```

Then:

- does the replica's `Relay_Master_Log_File` still exist on the source?
- is the replica already lagging or in error?

- does the binlog window cover the requested position?

If the answer is no, the button should not simply send `CHANGE MASTER`. It should show a warning and propose a rebuild.

Conclusion

Switching to GTID did not purge the source binlogs. But the `CHANGE MASTER` used to enter GTID and then return to file/position reset the replica's local relay logs.

In a healthy environment, the replica downloads them again from the source. In the incident analyzed here, the requested coordinate was already outside the source binlog window. Once the local relay logs had been discarded, there was no remaining source from which to replay the missing range.

The operational lesson is clear: before switching between GTID and file/position, validate the source binlog window. Otherwise, a reversible test becomes a mandatory resynchronization.