DR drill Time line id mismatch

Let’s break down what really happened behind the scenes in your case — step by step — so you’ll understand why timeline ID mismatch occurs even when you think you shut down the primary cleanly.

▯ Step-by-step of what actually happened during your drill

1▯ You stopped the primary — but it didn’t stop immediately

- You ran:

```
systemctl stop postgresql
```

or

```
pg_ctl stop -D /var/lib/pgsql/15/data
```

- but PostgreSQL took time or hung.
- That means:

1. PostgreSQL was still flushing WALs, writing checkpoints, or performing a shutdown checkpoint.
2. You may have forced it to stop later (e.g., pressing Ctrl+C, or kill -9, or running stop again).

- So the shutdown was not fully clean, even though you eventually got it to stop.
- That’s key — if PostgreSQL doesn’t write the “clean shutdown” marker into pg_control, the next time it starts, it performs crash recovery.

2▯ Meanwhile, you promoted the DR standby

- pg_ctl promote -D /var/lib/pgsql/15/data

or via Patroni/Repmgr.

- When this happened:

1. DR standby stopped following the old primary’s WAL stream.
2. It created a new timeline (Timeline +1).
3. A new file pg_wal/00000002.history was written.

- So now:

1. Old Primary = timeline 1
2. DR (new primary) = timeline 2

3� You restarted the old primary as a standby

- You changed:

```
standby.signal
primary_conninfo='host=DR_primary ...'
```

- and started it.
- But PostgreSQL said something like:

FATAL: requested timeline 2 is not a child of this server's history

- Why?

Because before promotion, the DR server and old primary had the same WAL history (TLI=1).

When you promoted DR, it created timeline 2 starting from an LSN slightly behind the old primary's last WAL (because old primary had some unflushed WALs when it stopped).

- So their WAL histories diverged, like this:

```
� Timeline illustration
Timeline 1 (before failover)
  Primary: ...A----B----C----D (crash while writing D)
  Standby: ...A----B----C
```

- Promotion (on standby)

Standby promoted → creates Timeline 2 starting from C

- Old Primary restart:

```
  Data ends at D (Timeline 1)
  Wants to follow Timeline 2 (starts from C)
  ❌Not possible — histories diverged
```

4. Why it shows timeline mismatch

- Because PostgreSQL tracks history strictly.
- When you promoted the DR, PostgreSQL wrote a file like:
- **pg_wal/00000002.history:**

```
1      0/50000000   still from timeline 1 until here
```

- But the old primary's last checkpoint LSN might be beyond 0/50000000, so it cannot logically attach to timeline 2.

- They’re two different branches of WAL history.

▯ 5▯ Root cause in your case

- ✅You thought you shut down the primary —
- ❌But it didn’t cleanly finish checkpointing, leaving some WALs unsynced.
- So your old primary had more data than the DR standby at the time of promotion.
- When DR became new primary → new timeline started before old primary’s last WAL record → divergence → timeline mismatch.

▯ 6▯ How to fix it safely

- ✅Use pg_rewind
- pg_rewind detects the common WAL point and removes diverged WAL/data pages from the old primary, aligning it to the DR’s new timeline.

```
● systemctl stop postgresql

pg_rewind --target-pgdata=/var/lib/pgsql/15/data \
  --source-server="host=<DR_primary_IP> user=postgres port=5432 dbname=postgres"
```

- Then reconfigure standby.signal and start again.

✅If pg_rewind fails or WALs missing:

- Take a new base backup from the new primary:
- pg_basebackup -h <DR_primary_IP> -U replica -D /var/lib/pgsql/15/data -Fp -Xs -P

7▯ Best Practices (to avoid this next DR drill)

Step	Action	Why
1	Always run clean stop: pg_ctl stop -m fast	Ensures full checkpoint
2	Check that service stopped completely before promotion	Avoid partial flush
3	Check replication lag before promoting	Avoid timeline split
4	Always use pg_rewind when re-adding old primary	Handles divergence safely
5	Monitor .history file after promotion	Confirms new timeline created

✓Summary (in simple terms)

- You saw timeline mismatch because:

1. Primary didn’t fully flush WAL before stop (dirty shutdown)
2. DR was promoted → new timeline created
3. Old primary had extra WAL data not present on DR
4. When it restarted, its history didn’t match the new primary’s

- Result:
→Timeline mismatch error
- Fix:
→Use pg_rewind before rejoining old primary