FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

RPO Zero Is a Network Question, Not a Backup Question — What "No Data Loss" Really Costs Between Two Sites

A presales war story about a revision request that read "RPO ≈ 0 and the lowest possible RTO" — the sentence everyone files under backup and almost nobody files under latency — and the four numbers I now collect before I promise any recovery point.

Daanial Farrukh · 04 November 2026 · https://quabin.com/rpo-zero-network-question-not-backup/

The first version of the proposal had been accepted in principle. Primary site in one city, a warm disaster-recovery site in another, the database replicated asynchronously, a recovery time of a few hours. Normal. Defensible. Then the revision request arrived.

"We now need RPO of approximately zero and the lowest RTO you can commit to. Please update the technical and financial proposals."

I joined the call with their database team already on it.

"Zero means zero," said their lead DBA. "Our team will check the design. Do not write 'near zero' and hope."

"Understood. Which component do you expect to deliver it?"

"Your backup. You proposed a backup service. Make it continuous."

"A backup cannot give you zero. It can give you a copy from a point in time. How far apart are your two sites, in milliseconds?"

Silence.

"And what is the peak redo rate of the database, in megabytes per second?"

"Why does that matter for a backup?"

That question — asked honestly, by a competent team — is the whole article.

The Trap, Named

That's the trap. **RPO** is a phrase from the backup world, so people hand it to the backup product. But a recovery point of zero cannot be produced by anything that copies data on a schedule, however short the schedule. It can only be produced by a system that refuses to confirm a write until a second site has it. That system is synchronous replication, and whether it is possible at all is decided by one network link

— its round-trip time and its bandwidth against the database's peak write rate.

RPO zero is not a feature you buy. It is a latency budget you either have or do not have.

What RPO Actually Means for Each Kind of Copy

Here's the part that trips people up. Three different things get called "a copy of the database", and each one carries a different recovery point.

A backup writes a restore point at intervals. Nightly full, hourly incremental, whatever the policy says. Your RPO is the gap between restore points — hours, sometimes a day. In the design I was defending, the backup tier was a hardened repository with a 7-day immutability window. Its job is to survive ransomware and human error. Its job is not to be current.

Asynchronous replication streams every change to the standby as fast as the link allows, but the primary commits first and ships later. The recovery point is the transport lag — normally seconds, sometimes a minute when the link is busy. The primary never waits.

Synchronous replication sends the change to the standby and waits for the acknowledgement before the primary tells the application "committed". The recovery point is zero. The cost is that every single commit now includes a round trip to the other city.

In arrow form, the synchronous path for one transaction looks like this:

Application commit → primary writes redo → redo crosses the link → standby writes it → acknowledgement crosses the link back → primary returns "committed" to the application

Two trips across the link, inside every commit, for as long as the database runs. If the round-trip time between the sites is 20 milliseconds, every commit grows by roughly 20 milliseconds. A busy order-entry screen that used to commit in 2 milliseconds now commits in 22. Nobody changed the application. The requirement did it.

Where this confusion is common:

- Tender requirement sheets, where "RPO" and "backup" sit in the same table row and get read as one thing.
- Marketing for "continuous data protection", which sounds like zero and usually means seconds.
- Database teams and infrastructure teams who each assume the other side owns the replication link.

Why Nobody Sees It in Testing

A disaster-recovery drill is the quietest moment in the database's year. It runs at night, on a weekend, with the application mostly idle. The redo rate is a trickle. The standby is "fully in sync" within a second of any change because there is almost nothing to send. Commit latency looks fine because almost nobody is committing.

Then the design goes live, month-end arrives, the redo rate climbs to its real peak, and one of two things happens. In asynchronous mode the transport lag stretches — your "seconds" become "a minute or two" exactly when the data matters most. In synchronous mode the application slows down across

the board, and if the link flaps, the primary stalls or drops back to asynchronous on its own. Either way, the drill never showed it, because the drill never carried load.

Absence of lag at 2 a.m. is not proof of RPO. It is proof that nothing was happening.

Why Replication Must Never Be Sold as Backup

This is the second thing the revision request nearly flattened, and the customer's own tender had a section demanding the opposite — a clear separation between backup and disaster recovery. Their DBA team would have caught any blurring, and rightly so.

Replication copies every change, including the bad ones. A dropped table arrives on the standby in seconds. An application bug that corrupts rows corrupts them in both cities. Ransomware that encrypts data through the application layer is replicated faithfully. The standby is a mirror, and a mirror reflects the damage.

The backup is the thing that does not follow. It holds restore points from before the mistake, in a repository the production credentials cannot modify, for a fixed retention window. In the design that stayed, the database was backed up natively by the database's own tooling to that repository, and the replication product was listed under "disaster recovery" with its own column. Two products, two columns, two jobs.

Plain takeaway: if a proposal has one line that claims to be both your backup and your zero-RPO replica, it is neither.

How to Actually Decide: The Four Numbers

Before anyone writes "RPO 0" into a commitment, collect these four numbers. None of them come from the backup product.

Step 1: Measure the round-trip time between the two sites.

This is read-only and takes a minute. From a server at the primary site:

```
ping -c 100 203.0.113.20
```

Read the last line. The average round-trip time is the cost you will add to every synchronous commit. The maximum tells you what a bad moment looks like. If you want the path as well as the number:

```
mtr -r -c 100 203.0.113.20
```

Expected output is a stable average with a small spread. A failure looks like an average that is fine and a maximum ten times higher — that spike is what a synchronous commit will wait for.

Step 2: Measure the peak redo rate, not the average.

The database already knows how much change it generates. For Oracle, the archived redo gives you a per-hour history over the last days:

```
SELECT TRUNC(completion_time,'HH24') hour, ROUND(SUM(blocks*block_size)/1024/1024) mb FROM
v$archived_log WHERE completion_time > SYSDATE-7 GROUP BY TRUNC(completion_time,'HH24') ORDER
BY 1;
```

Find the busiest hour, divide by 3,600, and you have peak redo in megabytes per second. That peak — not the daily average — is what the link must carry without the lag growing.

Step 3: Size the link against the peak, with headroom.

Oracle's own Data Guard guidance sizes the transport link by dividing the redo rate by 0.7 to leave room for protocol overhead, then converting to bits:

```
Required Mbps = ( peak_redo_bytes_per_second / 0.7 ) x 8 / 1,000,000
```

If the number comes out above the link you have, asynchronous lag will grow at peak and synchronous commits will queue. In the case that started this article, the peak redo rate was one of the three facts we had to request from the customer before the revision could be honest, together with the measured latency and the current link bandwidth. Nobody had them to hand. That is normal, and it is exactly why the question matters.

Step 4: Choose the protection mode and name who presses the button.

With the three numbers in front of you, the mode chooses itself. If the round-trip time is a few milliseconds and the link has headroom at peak, synchronous is affordable. If the sites are a city apart and the link is shared or thin, asynchronous with a committed lag ceiling is the honest answer. Read the current state from the database, not from the diagram:

```
SELECT name, value FROM v$dataguard_stats WHERE name IN ('transport lag','apply lag');
```

Transport lag is your real RPO right now. Apply lag is how far behind the standby is in applying what it already has. Run it during the busiest hour you found in Step 2, not on a Sunday.

Then write down who performs the failover. Automatic failover needs an observer process that watches both databases from a third location and makes the switch. If the customer's database team owns that observer — which is what we agreed, with the observer VM provided in a third data centre — then the recovery time splits into two owned parts: the provider's part (the standby platform is running and traffic is redirected) and the customer's part (the database role switch). Commit to the part you control. In our case that was 15 minutes for the platform and the traffic switch, and the commitment said so in plain words.

Step 5: Verify under load. This step is where most tutorials stop and where mine starts.

Run the drill with the application busy, or replay a representative load. Measure commit latency on the primary before and after switching the mode. Watch transport lag during the run:

```
SELECT name, value, time_computed FROM v$dataguard_stats WHERE name = 'transport lag';
```

If the lag climbs steadily during the test and only recovers when load stops, the link is undersized for that peak. If commit latency doubled when you enabled synchronous transport, that is not a bug. It is the verification — the round trip you measured in Step 1, showing up exactly where the arithmetic said it would.

What We Committed Instead of "Zero"

The honest version was shorter than the ambitious one, and the customer's database team accepted it.

- Replication stayed **asynchronous** over one dedicated link between the two sites. No extra relay hop, no second link.
- **RPO committed as under one minute** — typically seconds, explicitly "not zero" — because that is what the measured path could promise without slowing production.
- **RTO 15 minutes**, scoped to the provider's part: the disaster-recovery platform kept hot at full production size, and the traffic switch at the edge. The database failover belonged to the customer's observer.
- The **observer** ran on a VM in a third data centre, owned and configured by the customer's team, with the provider guiding.
- Backup stayed a **separate line** — native database backups into a hardened, immutable repository — and was never described as part of the recovery-point story.
- One **yearly drill**, under load, with the lag query above as the exit criterion.

Nothing in that list is exotic. All of it is measurable. That is the point.

The Bigger Lesson: Six Checks Before You Promise a Recovery Point

1. **You know the round-trip time between the sites.** ping -c 100 <standby-ip> — the average is your synchronous commit tax.
2. **You know the peak redo rate, by hour, for the last week.** The v$archived_log query above — size the link to the peak.
3. **The link has headroom at that peak.** Redo bytes per second ÷ 0.7 × 8 — if the result is above the link, lag will grow when it matters.
4. **The protection mode is stated in words a DBA can verify.** v$dataguard_stats during peak hour — transport lag is the real RPO.
5. **Failover has a named owner and a named observer location.** If the answer is "the platform handles it", ask which process, on which host, in which site.
6. **Backup and replication are separate lines with separate jobs.** One holds history behind immutability; the other mirrors the present. Never the same box.

What I Tell Every Customer Now

When a requirement says "RPO zero", I ask one question before I touch the proposal:

"What is the round-trip time between your two sites, and what is your peak redo rate?"

If they answer with two numbers, they are in good shape — the design is arithmetic from there. If the answer is "our backup vendor handles that", we're not done yet. The backup vendor was never asked that question either.

I write about the questions I ask in real customer rooms — the ones that change architectures and surface the problems no one else noticed. If you found this useful, drop your email below and I'll send the next one when it's ready.

FAQs

Can a backup product deliver RPO zero?

No. A backup writes restore points at intervals, so the recovery point is always the gap between the last restore point and the failure. Even "continuous" backup products capture changes with a small delay. Zero data loss needs synchronous replication, where the primary database waits for the standby to acknowledge each change before confirming the commit.

What does synchronous replication cost in practice?

Every commit waits for a round trip to the standby site. If the two sites are 20 milliseconds apart, every commit grows by roughly 20 milliseconds, all day, for every transaction. Measure the round-trip time with ping or mtr first; that number is the price of zero.

What is the difference between transport lag and apply lag in Oracle Data Guard?

Transport lag is how much redo the primary has generated that has not yet reached the standby — that is your real recovery point at this moment. Apply lag is how far the standby is behind in applying redo it already holds, which affects recovery time rather than data loss. Both come from the view v$dataguard_stats and should be read during peak hours.

Is Oracle Data Guard a backup?

No. Data Guard mirrors changes continuously, so a dropped table, a corrupting bug or ransomware acting through the application reaches the standby within seconds. A backup holds restore points from before the mistake, ideally in an immutable repository. Keep them as separate products with separate lines in any proposal or contract.

How do I size the network link for database replication?

Find the peak redo rate — the busiest hour from v$archived_log divided by 3,600 — not the daily average. Oracle's guidance divides that rate by 0.7 for overhead and converts to bits per second. If the result exceeds the link, asynchronous lag grows at peak and synchronous commits queue.

Who should own the failover in a provider-hosted DR design?

Name it explicitly. Automatic database failover needs an observer process in a third location; whoever configures and runs that observer owns the database part of the recovery time. The provider commits to its own part — the standby platform running and traffic redirected — and the agreement should state both parts and both owners.

Read online: https://quabin.com/rpo-zero-network-question-not-backup/