



Caller ID Delivery Analysis: RingCentral SIP Capture

Why the caller name & number are not reaching the POS — findings from the packet capture

Tech Support Analysis — 08JUN26 · Revised 10JUN26

Bottom line up front: Thanks for sending these over, and thanks for pushing me to look again. I went back through your raw Wireshark capture packet-by-packet. The good news is unchanged: the caller ID name and number **are present** in your call signaling, and that signaling is **not encrypted** — so encryption is not what is stopping the data. But here is the part I want to correct from my first pass: this setup **worked for years** on this same RingCentral account, through this same Vertex, and only stopped about a month ago. That history matters — it proves the caller ID format RingCentral sends has **always been readable** by your device. So this is not a case of “RingCentral can’t work with the Vertex.” Something **changed about a month ago** in how your calls are delivered: your phones are now talking to RingCentral’s cloud over **TCP** on their cloud ports, instead of the way they used to. That is why the caller ID stopped — and why the replacement unit you installed behaves the same way. This is a change-in-delivery problem we can fix, not a dead device and not a locked-down carrier.

21

Inbound call setups (INVITE) captured

100%

of the SIP signaling readable in cleartext

0

Encrypted (TLS) SIP packets found

TCP

Transport carrying the SIP signaling

1. What I analyzed

I’ve now gone through **two** raw captures from the site — the first one (21-230705-33-24Taketwo.cap , 1,865 packets) and a second one you sent so I could double-check the findings (21-240409-12-24_63 RingCentral.cap , 6,916 packets) — along with your Wireshark screenshots. I did not rely on the screenshots alone; I parsed both raw files directly and pulled apart the call-setup messages. The first capture holds **21 inbound call setups (INVITEs)** with the caller ID right in them; the second is mostly an outbound call plus phone registrations, and it **confirms** the same picture. Both files tell the same story, which I walk through below.

2. What the capture proves

Finding 1 — The carrier is RingCentral. Confirmed.

Your phones are registering to `sip.ringcentral.com` (the REGISTER / 200 OK exchanges at the top of the capture), which confirms RingCentral as your carrier.

Finding 2 — The SIP is NOT encrypted, and the caller ID is right there.

This is the important one. If RingCentral were encrypting the signaling (SIP over TLS), Wireshark could only show `Application Data` — it could never display words like `INVITE`, `NOTIFY`, or `200 OK`. But it does display them, and I confirmed it: I found **zero** encrypted/TLS SIP packets and read every SIP message in plain text. Every inbound call setup carries the caller's name and number in the clear. Here is a real example from the capture (caller details partly masked for privacy):

```
# Inbound call setup (INVITE) — from RingCentral cloud to a phone on the LAN
INVITE sip:1732679####@192.168.128.133:6216;transport=tcp SIP/2.0
From: "POCESTA P****" <sip:646403####@104.245.57.146>
To:   "POCESTA P****" <sip:1732679####@69.116.241.82>
Contact: <sip:646403####@104.245.57.146:5090;transport=tcp>
```

The caller name (`From` display name) and the caller's 10-digit number are both present and readable. **The data we need is being delivered.**

Finding 3 — Right now, the signaling rides on TCP, delivered cloud-direct to each phone.

In this capture, every one of the 21 call setups uses `transport=tcp` (not UDP), and your phones are even **registering to RingCentral over TCP**. RingCentral sends each call setup **straight from their cloud servers** (the `104.245.57.x` addresses) to an **individual phone** on your LAN (`192.168.128.x`) on a high, per-call cloud port — and nothing rides the standard SIP port your network used before. There is no on-premise phone system in the middle funneling calls through one predictable point. (Why that matters is the next section — this is almost certainly the thing that changed.)

Finding 4 — The capture is missing pieces of the TCP conversation.

Throughout the file Wireshark flags `[TCP ACKed unseen segment]` and `[TCP Spurious Retransmission]`. In plain terms: the device that recorded this capture **acknowledged receiving data it never actually saw** — the monitoring point was only catching part of the conversation. That matters enormously for TCP, because a SIP message sent over TCP can be split across several packets, and you must have **all** the pieces to rebuild the message and read the caller ID out of it.

Finding 5 — The “UDP” and the “TLSv1.2 (encrypted)” lines you can see are *not* your calls.

When you scroll the capture you'll notice two things that look like they might contradict the above — some `UDP` packets, and some `TLSv1.2 (encrypted)` packets. I checked both, and neither one is your call signaling:

- **The UDP packets are not SIP.** They are broadcasts to `255.255.255.255` on port `29810`, and the contents are an application status/“heartbeat” message in JSON format (you can see `"header":{"version"...` sitting in the packet bytes). That is a program on the network

announcing itself — not a phone call, and not the caller-ID format our device sends. So the caller ID is not hiding in the UDP.

- **The TLSv1.2 (encrypted) packets are the phone's directory/setup channel, not the calls.** Those packets are your Yealink phone talking to a server on TCP port 636 — a secure directory/provisioning channel (the phone pulling its configuration and contact list). Your *calls* go to a different RingCentral server (the 104.245.57.x addresses) on port 5090, and that call signaling is in **cleartext**, not TLS. They are two different conversations, to two different servers — easy to mix up because they share the same capture file.

Here is the tell that settles it: in your screenshot, Wireshark prints the call messages out in plain English — INVITE, REGISTER, NOTIFY, BYE, 200 OK, sip.ringcentral.com. It could **only** do that if those messages are unencrypted. If your calls were riding inside that TLS, every one of those lines would instead just read Application Data — unreadable. So the encrypted traffic is real, but it is not the calls; the calls are readable, and they are on TCP.

3. This setup worked for years — so something changed

This is the key correction to my first pass. You have had caller ID flowing into your POS on this same RingCentral service, through this same Vertex, for **years** — and it only stopped about a month ago. That history proves the caller ID format RingCentral sends has **always been readable** by your Vertex. So the problem is not that “RingCentral and the Vertex are incompatible.” Something about *how your calls are delivered* changed recently — and the capture I have is a snapshot taken **after** that change, so it shows the broken end-state, not the way it ran before.

The capture points straight at the most likely change: your phones are now **registering to RingCentral over TCP** and receiving every call over TCP from RingCentral's cloud, on high cloud ports — with nothing on the standard SIP port your network used before. The transport your phones use is normally driven by how RingCentral provisions the phones/account, which is why this most likely points to a change on the carrier side — though a local change (a new router or firewall, a SIP-ALG setting, or a phone firmware update) can also push phones from UDP onto TCP. The most likely explanation is that these phones moved from UDP to TCP signaling, and that switch is what took the caller ID off the path your Vertex has always watched.

The replacement unit is actually a useful clue. You bought a new Vertex and it behaved exactly the same way. That tells us the original unit was never the problem — a brand-new box failing identically means the cause is *upstream* of the device, in how the calls are delivered and mirrored to it. That is good news: it points us at a setting to change, not more hardware to buy.

4. About the "encrypted SIP" concern (this is the good news)

You might reasonably expect RingCentral to encrypt their SIP signaling — many carriers do, and that was the natural first thing to suspect. On your account, though, they don't: the

signaling is plain, readable text and the caller ID is sitting right inside it, exactly as shown above. So encryption is not what's blocking your POS.

Why this is good news for you: an encrypted carrier would be a dead end. A transport-and-visibility issue is something we can correct together. The caller ID is being handed to your network in the clear — we just have to make sure the Vertex receives a *complete* copy of it.

5. Why the caller ID is not reaching the POS right now

The Vertex reads caller ID by watching the call-setup (INVITE) message go by and pulling the name and number out of it. For that to work, two things have to be true. Based on the capture, here is where it is breaking:

Requirement	What the capture shows	Impact
The call setup must reach the Vertex the way it did before (almost certainly UDP SIP on your network)	100% of the signaling is now <code>transport=tcp</code> , delivered from RingCentral's cloud on high per-call ports — there is no UDP SIP on the line anymore, and nothing on the standard SIP port.	This is the change from how it ran for years. The call setups your Vertex has always read are no longer crossing your network in that form, so it has nothing to extract. Restoring UDP SIP delivery restores what the device reads.
It must see a complete copy of the call setup	The monitoring point was missing TCP segments (<code>ACKed</code> unseen segment everywhere), and calls now go cloud→phone as switched, per-call traffic.	Even apart from the transport change, a Vertex that only gets part of the traffic can't read a clean call setup. The switch mirror/SPAN needs confirming regardless.

In short: the caller ID is in the air, unencrypted — but it is now arriving as **SIP-over-TCP from RingCentral's cloud**, which is a change from how your calls used to reach the Vertex, and the monitoring point is **not seeing a complete copy** of it either. That is why it stopped populating into the POS.

6. What I recommend we do next

- 1. First, the cheapest thing to try — ask RingCentral to put these phones back on UDP.** The capture's clearest lead is that your phones are now registering and taking calls over **TCP** to RingCentral's cloud (port `5090`), and that transport is normally controlled by how RingCentral provisions your phones/account. Ask them to set these phones (or the account) back to **UDP** SIP signaling. If they can, that most likely restores exactly the delivery that fed your POS for years — on the hardware you already have. *Honest caveat:* some carriers have standardized on TCP and no longer offer UDP. If that turns out to be the case here, it is not a dead end — see the on-site converter option below.

2. **Make sure the Vertex sees a complete, two-way copy of the traffic.** The Vertex has to be on a properly configured switch mirror/SPAN port (or inline) that mirrors *both directions* for every phone, without dropping packets — this capture showed packets being missed at the monitoring point, so this needs confirming regardless of the transport.
3. **Then re-capture to validate.** Take a fresh capture from the Vertex's exact connection point. If it shows the call setups arriving as UDP SIP and is free of `ACKed unseen segment` warnings, the device is getting what it needs again.
4. **If RingCentral won't move you back to UDP — put a small SIP converter on site.** The idea is to give the Vertex back the UDP signaling it reads, locally, without depending on RingCentral. A small on-premise SIP device (a “session border controller,” or SBC) sits on your network: your phones talk to *it* over **UDP** on the LAN — which is the leg the Vertex taps and reads — while it carries the calls up to the carrier over **TCP**. In effect it translates the transport so your existing Vertex keeps working. Options range from a software proxy (Kamailio / OpenSIPS, or a small FreePBX/Asterisk box) on an inexpensive mini-PC, up to a turnkey hardware SBC (AudioCodes Mediant, Patton SmartNode, Sangoma, or a Grandstream UCM). *One important caveat:* because RingCentral is a hosted service, routing the phones through a local box usually needs RingCentral's cooperation (their “bring-your-own-device / SIP-trunk” path) and can affect some cloud features — so we would confirm that with them before buying anything.
5. **Or, if you'd rather not touch the call path at all:** a newer Vertex platform that reads SIP-over-TCP directly is the other route — it keeps the same passive, plug-in install you have today. Tell me which direction you prefer (carrier UDP, on-site converter, or TCP-capable Vertex) and I'll come back with exact models and pricing. I'd still try the free UDP ask first, because — as the replacement unit showed — the fix here isn't automatically more hardware.

What I need from you to wrap this up: two things. First, anything that changed around the time it stopped about a month ago — a RingCentral phone or firmware update, new phones, a switch or network change, anything at all. Second, exactly where the Vertex is connected on your network (which switch port, and whether port mirroring is set up), plus the Vertex model and firmware version on site. With the proof above that your caller ID is unencrypted and already present, those answers will let me pinpoint what changed and get the caller name and number flowing into your POS again.