



Programming an SBC / Firewall to Convert Carrier TCP SIP → UDP for the Vertex

So a legacy Whozz Calling Vertex can read caller ID on RingCentral and other carriers that signal SIP over TCP or encrypted (TLS) — using an Edgewater (or equivalent) SBC

Installation & Configuration Guide — 24JUN26

The whole idea in one breath: the legacy Vertex reads caller ID by *passively watching* the call-setup messages (SIP INVITE s) cross the LAN, and it is built to read those when they arrive as **plain UDP**. Carriers like **RingCentral** now hand their phones SIP over **TCP** (and often **encrypted/TLS**) — which the legacy box cannot follow. The fix on this page is to put a **Session Border Controller (SBC)** — an Edgewater is one — in the call path. The SBC talks to the carrier however the carrier wants (TCP/TLS), then **re-creates the same calls as plain UDP SIP toward the phones on the LAN**. That UDP leg is the part the Vertex taps, and it can read it. This is a real protocol conversion done by a real device — not a setting you flip on the Vertex.

1. Why this is needed

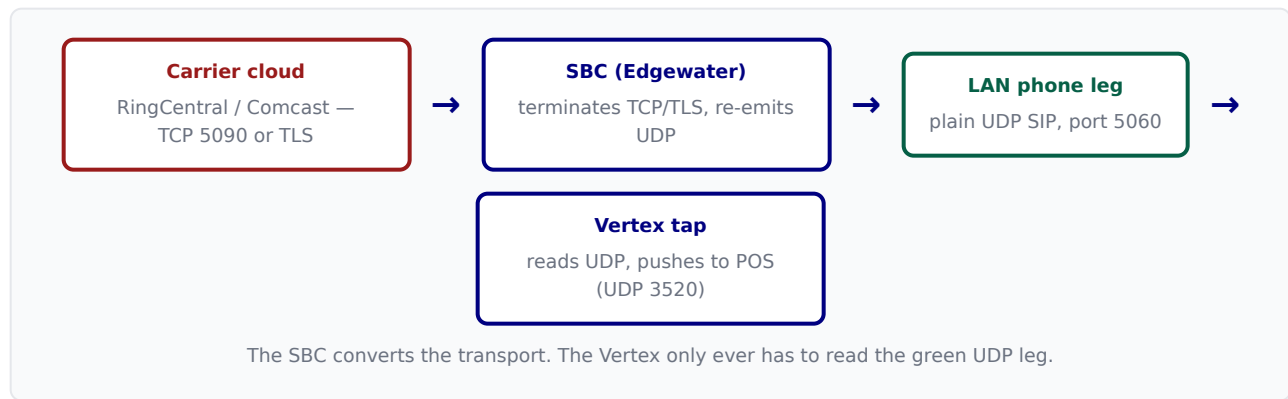
The Vertex is a **passive tap**. It sits in-line on its two VoIP Bridge Ports, watches every packet that crosses, and pulls the caller's name and number out of each call-setup **INVITE**. It works cleanly when that **INVITE** arrives as a **UDP** datagram — one self-contained packet it can grab and read. That is how Whozz Calling sites ran for years.

Two carrier changes break that passive read, and an SBC fixes both:

- **Encrypted SIP (TLS).** If the carrier encrypts the signaling, no passive device can read it — the words **INVITE** / **200 OK** never appear, only ciphertext. (This is the Comcast Business Edge case in the original note: encrypted on the carrier side.)
- **SIP over TCP.** RingCentral delivers its call setups over **TCP** (on RingCentral's outbound-proxy port **5090**), not UDP. TCP is a *stream* — one SIP message can be split across several packets that must be reassembled in order first. The legacy hardware was never built to do that, so on a TCP-only site its caller ID is unreliable (reads some calls, misses others).

An SBC solves both at once. It *terminates* the carrier's encrypted/TCP signaling on the WAN side and *re-originates* brand-new, plain **UDP** SIP toward the LAN phones. Whatever the carrier does on its side, the Vertex gets handed the unencrypted UDP it understands.

2. The fix at a glance



The one rule that makes it work: set the SBC's **LAN / phone-facing** SIP transport to **UDP**. Leave the **carrier / WAN-facing** side on whatever RingCentral requires (TCP/TLS). The Vertex only ever needs to see the UDP leg.

3. Before you start — what you need

- **An SBC that can do transport interworking (a B2BUA).** The **Edgewater / EdgeMarc** in the original setup is one. So are AudioCodes and Ribbon hardware SBCs, or a software SBC (FreePBX/Asterisk, Kamailio, OpenSIPS). It must be able to register/trunk to the carrier on one transport and present the phones a *different* transport — not just pass packets through.
- **Carrier permission to put an SBC in front of the phones.** Because RingCentral is a hosted service, re-homing the phones onto your SBC normally needs their **Bring-Your-Own-Carrier (BYOC) / SIP-trunk / manual-provisioning** arrangement. Good news: in those modes RingCentral *does* support plain **UDP SIP on port 5060** — TLS/TCP is only the default for their auto-provisioned hosted desk phones. **Confirm this with RingCentral before buying or cutting over.**
- **The Vertex / Whozz Calling unit** and, if the phones are on their own network, a small **extra switch** (PoE if the phones need power — TP-Link is the easy choice).
- **A laptop with Wireshark** for the 60-second verification in Section 6.

What does NOT work — don't waste a site visit on these. You cannot make the Vertex "read TCP" with a DIP switch or a config tool — there is no such setting. You also cannot just *relabel* the SIP from TCP to UDP (e.g., rewriting the `Via` header): the label does not move the bytes off the TCP connection, and the call breaks. Converting the transport for real requires a device that terminates TCP and re-sends as UDP — that device is the SBC. (If the carrier will not allow an SBC and will not provision UDP, the remaining path is the TCP-capable **Vertex Quantum** platform.)

4. Program the SBC (the conversion itself)

Menu names differ a little by SBC brand and firmware, so these steps are written by *function*. On an Edgewater/EdgeMarc you'll find them under the VoIP / SIP sections of the web admin; the same logic applies to any SBC.

1 Log in to the SBC's web admin.

Browse to the SBC's LAN address (for an EdgeMarc, its LAN IP in a browser) and sign in as administrator.

2 Set up the carrier (WAN) side the way the carrier wants.

Configure the trunk/registration to the carrier using the transport **they** require — for RingCentral BYOC/trunk that is typically **TLS or TCP**, registering to their proxy (devices send to outbound-proxy port 5090 ; the SIP port is 5060). Use the trunk credentials RingCentral gives you. This side is allowed to be encrypted/TCP — the Vertex never reads it.

3 Set the LAN / phone side to UDP. (*This is the step that makes caller ID work.*)

In the SBC's local/LAN SIP settings — the side the phones register to — set the **SIP transport to UDP** on port **5060**. Make sure TCP/TLS is *not* being forced toward the phones. After this, the SBC hands the LAN plain, unencrypted UDP SIP.

4 Confirm the SBC is in back-to-back (B2BUA) mode, not transparent pass-through.

The SBC must **re-originate** the calls (act as a back-to-back user agent), not just NAT/forward the carrier's packets untouched. Only a re-originated call gets rebuilt as fresh UDP on the LAN. On most SBCs this is the default when phones register locally to the device; just verify it.

5 Point the phones at the SBC (over UDP), not straight at the carrier cloud.

The phones should register to the **SBC's LAN address using UDP**, so their calls ride the local UDP leg the Vertex can read — instead of going TCP/TLS direct to RingCentral's cloud.

Transport map — what each leg should be

Leg	Transport	Port	Who reads it
Carrier / WAN side (SBC ↔ RingCentral cloud)	TLS or TCP (carrier's choice)	5090 outbound proxy / 5060 SIP	The SBC only — never the Vertex
LAN / phone side (SBC ↔ phones)	UDP	5060	The Vertex taps this leg
Vertex → POS PC (caller-ID delivery)	UDP datagram (output)	3520	Your POS software

Don't mix up the ports. 5090 is RingCentral's cloud proxy (carrier side). 5060 is the standard SIP port — the UDP LAN leg the Vertex reads. 3520 is the Vertex's *output* port to the POS PC; it has nothing to do with SIP capture. (If caller ID reads but the POS shows nothing, that's a 3520 delivery / firewall problem, not a SIP problem.)

5. Place the Vertex on the (now-UDP) LAN leg

With the SBC handing out UDP on the LAN, drop the Vertex into that segment exactly as a normal install:

- **Add a switch for the phones if they're on their own network.** Run a cable from the Edgewater to a new switch (PoE if the phones need power; TP-Link is the easy pick), then land all the phone lines on that switch.
- **VoIP Bridge Ports (toward gateway):** connect the Vertex's VoIP Bridge Ports in-line between the main switch and the phone switch, so all the VoIP traffic crosses the Vertex.
- **To Phones port:** connect to the auxiliary switch where the VoIP phones live.
- **Management Port:** connect to the network — this is how you configure the unit and how it delivers caller ID (UDP 3520) to the POS PC.
- **Static IP:** the Whozz Calling unit uses a static IP. Set it to fit the network with the **Ethernet Link Config (ELConfig)** tool, which also sets options like duplicate-record handling.

6. Verify it worked (about 60 seconds)

Before you leave, prove the LAN leg is now readable UDP — don't rely on "a name popped once."

1. Put a laptop on the LAN phone segment (a switch mirror/SPAN, or in-line) and open Wireshark.
2. Make a test call and apply the filter `udp.port == 5060 || sip`.
3. **You want to see INVITE / 200 OK in plain English riding UDP 5060.** That means the SBC is presenting readable UDP — the Vertex will read it too.

If you still only see TCP 5090 (or encrypted "Application Data") on the LAN, the SBC is not presenting UDP to the phones — go back to Section 4 Step 3 and force the LAN-side transport to UDP, and confirm the phones are registering to the SBC and not straight to the carrier cloud. Until the LAN leg is UDP, the Vertex cannot read it.

7. Troubleshooting

- **Caller ID still blank.** Re-run the Section 6 check. If the LAN leg is UDP and readable, the issue is downstream: check the POS's caller-ID setup (ESC) and that the **UDP client port matches the Vertex output (3520)**, and that no firewall is blocking that port between the Vertex and the POS PC.
- **Amber link-light issues / unstable link (the original site's symptom).** Three amber lights do *not* mean "three devices attached" — amber here usually means the link dropped to a low-power mode (**Energy Efficient Ethernet**). Putting a **cheap unmanaged switch in-line** defeats EEE and steadies the link. (See the Vertex EEE customer solution note.)
- **DIP switches.** If behavior is odd, reset the Vertex DIP switches to their default positions.
- **"It works sometimes" with NO SBC in place.** That is the TCP transport mismatch, not a fix — treat intermittent caller ID as "not working." The reliable answers are: provision the carrier for UDP, put the SBC in front (this guide), or move to the TCP-capable Vertex Quantum.

Reference material

- Vertex Caller ID Installation Guide — central.toasttab.com/s/article/Vertex-Caller-ID-Installation-Guide
- CallerID.com installation diagrams — callerid.com/installation_diagrams.php
- Vertex Manual V2 — callerid.com/downloads/manuals/Vertex_Manual_V2.pdf
- Deluxe Ethernet Install Guide (ELConfig) — callerid.com/downloads/guides/IG_Deluxe_Ethernet_V4.1.pdf
- Companion bulletin: Legacy Vertex on a RingCentral (TCP) Site — [Tech-Bulletin-Legacy-Vertex-RingCentral-TCP-CallerID-24JUN26](#)