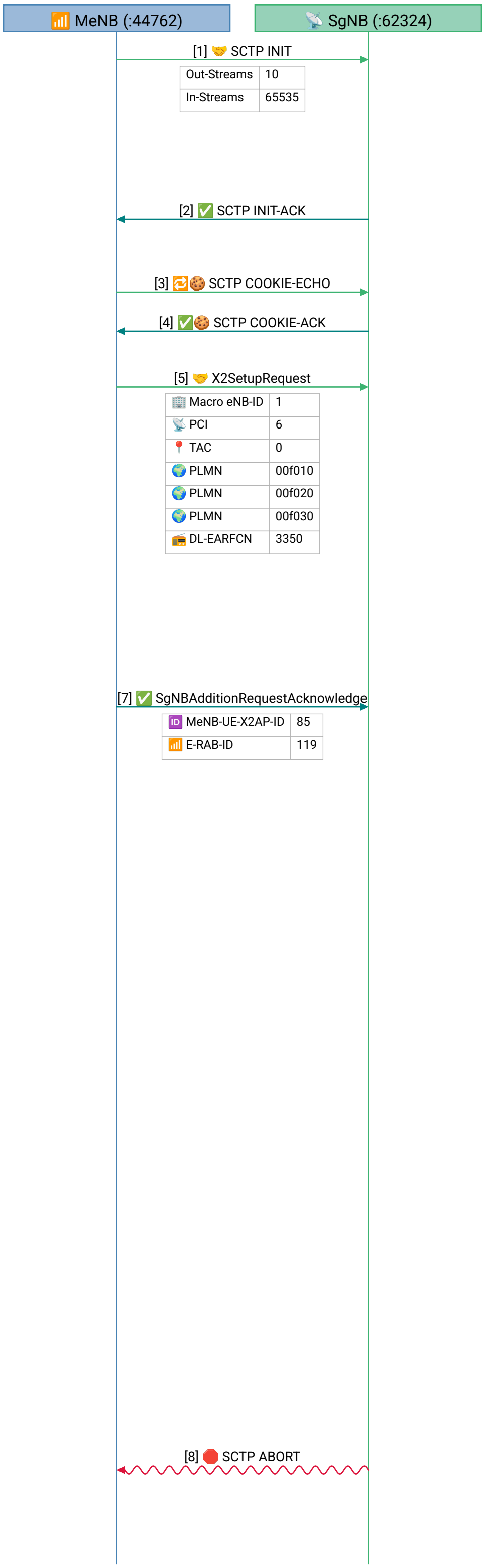


[Frame 1] SCTP INIT — the transport comes up first (again). As in the Request capture, X2AP rides on SCTP, so the two nodes build an association before any X2AP flows. The initiator on port **44762** — the master eNB (MeNB) side — offers **10 outbound streams** and accepts up to 65535 inbound. The multiple streams keep non-UE-associated signalling (the X2 Setup in frame 5) off the same stream as the UE-associated Acknowledge (frame 7). Watch the `44762 <-> 62324` port pair — it is the only thing distinguishing MeNB from the secondary gNB in this loopback trace. (The Request companion used `41471 <-> 62324`; the SgNB responder port is the same, the initiator port is not — one more sign these are two separate captures.)

[Frame 2] INIT-ACK. The responder on port **62324** — the secondary gNB (SgNB) side — accepts and returns its own parameters plus a state cookie, committing no association state yet (SCTP's anti-DoS design: the cookie carries the state, so the responder holds nothing until the echo returns in frame 3).

[Frame 3] COOKIE-ECHO. The MeNB echoes the cookie back, proving the round-trip — the third leg of SCTP's four-way handshake.

[Frame 4] COOKIE-ACK — association established. The gNB acknowledges; the association is open in both directions and ready to carry X2AP. The next two application frames are the X2 Setup and, this time, the Acknowledge.

[Frame 5] X2 Setup Request — bring the X2 interface up (non-UE-associated). The MeNB introduces itself and its served cell before any per-UE work. This capture advertises the *same* served-cell configuration as the Request companion — the same builder produced both:

IE	Value	Meaning
Global eNB-ID	**1**	macro eNB-ID (20-bit), decimal 1
Served cells	**1**	one E-UTRA cell advertised
PCI	**6**	physical cell ID
TAC	**0**	tracking area code
Broadcast PLMNs	**001-01, 001-02, 001-03**	the three PLMNs the cell broadcasts (00f0x0 encoding)
UL/DL EARFCN	**3350**	carrier frequency (same up and down in the test data)

As in the companion, no X2 Setup Response arrives — the builder fires the next PDU immediately and then aborts (frame 8).

[Frame 7] SgNB Addition Request Acknowledge — the reply the Request capture never got. This is the whole point of this companion: X2AP procedure **id-sgNBAdditionPreparation (27)** as a `successfulOutcome`. Where the Request carries the MeNB's *ask*, this carries the SgNB's *admit* — the SCG resources the gNB allocated and the RRC container the MeNB will forward to the UE.

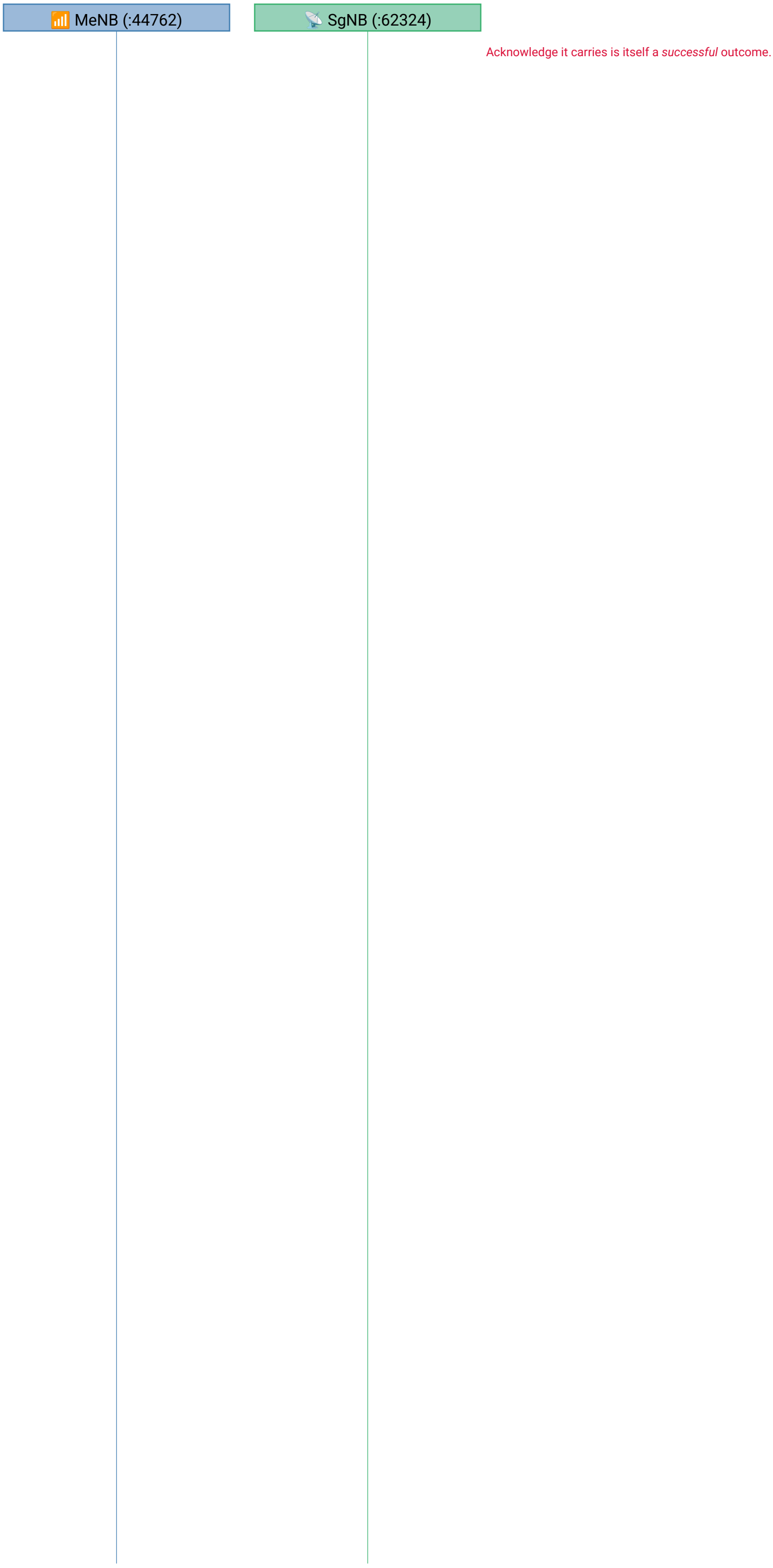
Verified contents (all in this frame):

IE	Value	Meaning
MeNB-UE-X2AP-ID	**85**	the MeNB's UE-association ID (echoed back)
SgNB-UE-X2AP-ID	**102**	the gNB now mints its *own* UE ID — the pairing the Request lacked
E-RAB-ID (admitted)	**119**	one E-RAB admitted, in E-RABs-Admitted-ToBeAdded-SgNBAddReqAckList
en-DC--Re-sour-...	pDCPatSgNB, mCGresources, sCGresources all **present**	the split-bearer shape, echoed from the request
re-sour-ce-configura...	sgNBPDCPnotpresent	PDCP stays at the **MeNB** — an MCG-anchored split bearer
sgNB-DL-GTP-TEIDatSCG	**0.0.0.0 / TEID 0x00000000**	the SgNB's downlink X2-U endpoint for the SCG leg (placeholder)

Two things make this the interesting half of the procedure:

- **The UE-association is now complete.** The Request carried only a MeNB-UE-X2AP-ID; here the gNB returns that *plus* its own **SgNB-UE-X2AP-ID (102).** That pair is the X2AP UE-association both nodes use for every later message about this UE. (Note the IDs — 85/102/119 — do not match the Request capture's 1/-/11: this is a different UE on a different association, not the literal answer to that request.)
- **The** `SgNBtoMeNBContainer` **carries an RRC** `CG-Config`. IE id-211, a 10-byte stub that Wireshark dissects as `nr-rrc CG-Config -> criticalExtensions: c1: cg-Config,` with every optional bit **false** (no `scg-CellGroupConfig,` no `scg-RB-Config,` no `configRestrictModReq)` — a well-formed but *empty* CG-Config. In a live add this container would hold the SCG `RRCReconfiguration` the MeNB relays to the UE to bring up the NR leg. This field is exactly what WS issue #14940 was about: the SgNB->MeNB container must decode as **CG-Config** (TS 38.331), not the `CG-ConfigInfo` that travels the *other* way in the Request's `MeNBtoSgNBContainer`. Read the direction carefully: the builder emits this PDU from the initiator socket (44762), so on the wire it flows toward the SgNB. Semantically it is the **SgNB's reply to the MeNB** — read it for the content it carries, not its on-wire direction.

[Frame 8] SCTP ABORT — teardown before a graceful close. As in the Request companion, the association ends not with an X2AP follow-up or an SCTP SHUTDOWN but with an SCTP ABORT, cause "**User initiated ABORT" (0x000c),** from the gNB-side endpoint (port 62324). It is a graceful, application-driven socket teardown — **not** an X2AP error — the builder simply closing the association after emitting its PDUs. So this capture, too, ends `incomplete::no-close` at X2AP and `failure::aborted` at SCTP, even though the


MeNB (:44762)
SgNB (:62324)
Acknowledge it carries is itself a *successful* outcome.