



PUBLIC-SECTOR WHITEPAPER

Institutional emergency-management stakeholders

Crisis Connect

Offline-First Emergency Communication for Disaster Operations

Authority review edition

Decision support for controlled pilot evaluation and staged public-launch planning

Auralis Industries
July 17, 2026

Edition Snapshot

Version	2.0.0
Date	July 17, 2026
Status	Authority Review Edition
Baseline	Crisis Connect Android v1.1.7 (versionCode 57) source baseline
Parallel iOS line	Crisis Connect iOS 1.1.2 store-listed parity implementation baseline
Audience	Emergency-management authorities, field-operations leaders, and technical reviewers
Launch posture	Authority-led, evidence-gated rollout on the Android v1.1.7 hybrid (offline-first + internet-augmented) line, with iOS 1.1.2 as the store-listed parity baseline under CI/TestFlight governance and Crisis Connect Web 0.1.0 as the reviewed operations and automation control-plane baseline

BUILDS

ANDROID 1.1.7

iOS 1.1.2

POSTURE

OFFLINE-FIRST

EVIDENCE-GATED

Document Purpose and Reader Guide

Document element	Current statement
Purpose	Decision support for controlled pilot evaluation and staged public-launch planning.
Intended audience	Emergency-management authorities, field-operations leaders, and technical reviewers.
Decision use	Go/No-Go review, launch planning, technical assurance alignment, and pilot governance discussion.
Not this document	Not a marketing brochure, not a nationwide emergency-network claim, not a certification record, and not a substitute for local field validation.
Reviewed build line	Android v1.1.7 (versionCode 57) is the current institutional deployment baseline and is publicly distributed through the Play Store launch workflow. The standalone iOS 1.1.2 client (iOS 17+, iPhone and iPad) is carried as the store-listed parity implementation baseline with an active Codemagic CI/TestFlight pipeline. The Crisis Connect Web 0.1.0 workspace remains the reviewed operations and self-hosted control-plane baseline; its release identity is carried by image tag, build SHA, and signed update manifests rather than by the static package version.
Revision scope	This July 17, 2026 edition is a major revision. It documents the hybrid transport architecture introduced on the 1.1.x line: an internet-augmented end-to-end encrypted messaging and calling plane (ECIES envelope and Signal-protocol forward secrecy) that operates alongside the unchanged offline Bluetooth incident path, plus authority channel communications, SOS cloud uplink, Crisis Sentinel on-device/online AI, mesh accelerators and push-to-talk, SPAKE2 contact bootstrap, 19-language localization, and the July 2026 web control-plane expansion (secure messaging and SFU/MLS group calls, external assistant channels, workflow scheduler, MCP exposure). Trust-boundary and privacy language is rewritten accordingly. Field-evidence snapshots from March–April 2026 are retained unchanged; the new internet-plane capabilities are documented as source-reviewed implementation baselines with their own open evidence gates.

Document element	Current statement
Reader guide	Decision boards, public authorities, and sponsor stakeholders can read the whitepaper through the decision conclusion and rollout guidance. Security reviewers, engineering assessors, and pilot teams should continue into the Technical Assurance Appendix.
Decision area	Executive takeaway
Mission fit	Bounded local coordination, responder interaction, and nearby operational continuity when internet and cellular infrastructure are unavailable or overloaded, with an internet-augmented end-to-end encrypted path that extends reach while infrastructure survives and degrades gracefully back to Bluetooth-only operation.
Deployment fit	Best suited to controlled pilots, trained responder teams, pre-provisioned organizations, and staged public-launch preparation rather than unmanaged mass-public deployment.
Security posture	Designated secure paths are protected with authenticated encryption and explicit trust gates; the internet messaging path adds Signal-protocol forward secrecy with the relay infrastructure holding ciphertext only; the Public GATT channel remains intentionally opt-in and non-confidential.
Non-goals that stay explicit	No replacement for 112/911 or equivalent emergency systems, no universal reach guarantee, and no unbounded anonymous store-and-forward network claim.
Launch posture	Authority-led, evidence-gated rollout on the Android v1.1.7 hybrid (offline-first + internet-augmented) line, with iOS 1.1.2 as the store-listed parity baseline under CI/TestFlight governance and Crisis Connect Web 0.1.0 as the reviewed operations and automation control-plane baseline. Release decisions should remain tied to measured acceptance gates and reviewer-visible evidence.
Recommended next step	Closed responder pilot, then mixed controlled exercise, then conditional broader deployment if operational and governance gates are met.

Executive Summary

Decision context. Crisis Connect addresses the failure window in which internet and cellular infrastructure is unavailable but nearby smartphones remain operational, and now also the surrounding periods in which infrastructure partially survives. This edition is written for emergency authorities and institutional reviewers who need a clear deployment position, not a product brochure.

- **Mission fit.** In earthquakes, infrastructure collapse, and off-grid incidents, the system is positioned as a bounded local coordination layer when wide-area communication fails first. The 1.1.x line extends the same contact graph across surviving infrastructure

so that pre-incident, degraded, and post-incident phases use one continuous communication surface.

- **Deployment model (hybrid, offline-first).** The offline incident path is unchanged: nearby Bluetooth device-to-device communication with a responder-only secure mesh extension and an opt-in Public GATT coordination channel under explicit policy controls. The v1.1.7 line adds an internet transport as a third drain of the same store-and-forward message queue: when validated connectivity exists, queued messages for internet-capable contacts are relayed as end-to-end encrypted envelopes through backend infrastructure that can route but cannot decrypt. A conversation started over Bluetooth continues over the internet and vice versa.
- **Operational posture.** The default civilian emergency posture remains low-power BLE/GATT standby for SOS discovery and responder-initiated contact. Public GATT relay is not framed as an always-on default for all users, and Wi-Fi Aware mesh remains a role-gated responder feature for time-bounded coordination. Internet-path failure degrades gracefully to Bluetooth-only operation without affecting the offline path.
- **Security and trust.** Designated secure paths use authenticated encryption and explicit trust promotion. The internet messaging path uses an authenticated ECIES envelope profile and, on current builds, Signal-protocol sessions (PQXDH post-quantum key agreement plus Double Ratchet forward secrecy) with safety-number verification and anti-downgrade pinning. Public GATT remains a broad-coordination feature, not a confidential responder channel.
- **Operational boundaries (restated).** The offline local incident path has no cloud dependency. On the internet path, cloud services relay ciphertext-only message envelopes, deliver push wake-ups, and support trust, telemetry, and SOS-uplink workflows; they never hold plaintext emergency content. This is a material boundary change from earlier editions, which excluded cloud from message relay entirely, and it is treated as such in the trust-boundary, privacy, and evidence sections.
- **Operational dashboard.** The reviewed Crisis Connect Web 0.1.0 line is documented as an authority-facing operations control plane covering panel administration, responder telemetry, SOS signal observations, audit visibility, licensing, update posture, SSO/session governance, and deployment health. The July 2026 line additionally carries an end-to-end encrypted messaging surface, SFU group calls with MLS media encryption, external assistant channels (Telegram/Slack), and a hardened workflow automation plane with scheduling and MCP exposure.
- **Launch posture.** Android v1.1.7 is the current reviewed deployment baseline and public store line. The iOS client line is version 1.1.2 for iOS 17+ on iPhone and iPad, developed as an explicit Android-parity program under an active CI/TestFlight pipeline; institutional go/no-go scope should still remain tied to Android until iOS evidence gates close.

- **Decision recommendation.** The product is suitable for staged authority-led rollout beginning with a closed responder pilot and progressing only through evidence-backed operational gates. The new internet plane is source-reviewed but has not yet accumulated the field-evidence depth of the offline path; pilots should gate on both.

Reader navigation. The opening whitepaper section is optimized for decision boards, public-authority reviewers, and pilot sponsors who need scope, operating model, deployment fit, and launch implications quickly. Detailed governance language, protocol profiles, cryptographic assumptions, privacy controls, and reviewer evidence materials are grouped later in the Technical Assurance Appendix.

Platform line	Current label	Current board reading
Android v1.1.7	Validated deployment baseline; public store line	Institutional go/no-go remains anchored to the Android reviewer packet. Offline-path field evidence carries forward from the March 31, 2026 set; the 1.1.x hybrid additions (internet E2E messaging/calls, authority channels, SOS uplink, Crisis Sentinel, mesh accelerators/PTT) are source-reviewed and carry their own open evidence gates.
iOS 1.1.2	Store-listed parity implementation baseline under CI/TestFlight	Developed as an explicit Android-parity program (wire and crypto byte-compatibility enforced by shared golden vectors) with a Codemagic CI pipeline publishing to TestFlight. Cross-platform messaging, QR pairing, and Bluetooth voice-call interop have developer-recorded checks against the Android 1.1.7 store build, but reviewer-grade field-depth and continuity gates remain open, and the newest feature line has not yet passed through the release pipeline.
Crisis Connect Web 0.1.0	Reviewed operations dashboard and self-hosted control-plane baseline	Next.js authority dashboard with Firebase cloud mode and Postgres/Keycloak on-prem mode is documented for panel operations, responder/signal telemetry, SSO/session governance, license and update controls, audit records, and installation-health oversight. The July 2026 line adds an E2E messaging surface, SFU/MLS group calls, Telegram/Slack assistant channels, a workflow scheduler, and MCP workflow exposure. Plaintext incident content remains outside the control plane.

Kısa Kurumsal Özet (TR – Bilgilendirici)

Not. Bu bölüm, kurumsal karar vericilere yönelik Türkçe bir bilgilendirme özeti sunar. Normatif gereksinimler, tehdit varsayımları ve kabul kriterleri yalnızca İngilizce ana metin üzerinden değerlendirilmelidir.

Başlık	Kısa açıklama
Amaç	Crisis Connect, internet veya hücresel altyapının devre dışı kaldığı durumlarda birbirine yakın cihazlar arasında çevrimdışı ve yerel iletişim sağlar; altyapının ayakta kaldığı dönemlerde ise aynı kişi listesi üzerinden uçtan uca şifreli internet mesajlaşması ve sesli/görüntülü arama sunar. Bağlantı kesildiğinde sistem kesintisiz olarak yalnızca-Bluetooth çalışmaya döner.
Önerilen kullanım	Sistem; kamu otoriteleri ve kurumsal ekipler için kontrollü pilotlar, saha tatbikatları ve kademeli kamuya açılış senaryolarında yardımcı yerel koordinasyon katmanı olarak değerlendirilmelidir.
Mevcut kapsam	Çevrimdışı acil durum yolu değişmemiştir: Bluetooth üzerinden cihazdan cihaza akış, yetkili ekipler için rol kontrollü Wi-Fi Aware güvenli mesh uzantısı ve isteğe bağlı Public GATT genel koordinasyon kanalı. Android v1.1.7 hattı buna ek olarak, aynı mesaj kuyruğunu doğrulanmış internet bağlantısı üzerinden boşaltan üçüncü bir taşıyıcı içerir: mesajlar uçtan uca şifreli zarflar hâlinde iletilir ve sunucular yalnızca şifreli metni yönlendirir, içeriği açamaz. Yakındaki kişiler, telefon numarası havada hiçbir türevle yayınlanmadan SPAKE2 tabanlı bir el sıkışmayla numara üzerinden de eklenebilir; bu özellik varsayılan olarak kapalıdır. Ayrıca kurum kanalı mesajlaşması, SOS bulut bildirimi, cihaz-üstü/çevrimiçi Crisis Sentinel yapay zekâ asistanı ve 19 dil desteği eklenmiştir.
Operasyon postürü	Varsayılan sivil acil durum postürü, düşük güçte BLE/GATT bekleme ile SOS yayını ve müdahale ekibi tarafından başlatılan temastır. Public GATT relay her kullanıcı için sürekli açık varsayılan mod olarak önerilmez; Wi-Fi Aware mesh ise zaman sınırlı ve rol kontrollü müdahale-ekibi kullanımına ayrılır.
Güvenlik sınırı	Güvenli iletişim yolları; QR ile eşleştirilen kişi sohbetlerini, Rescue güvenli sohbetini, yetkili müdahale ekiplerine yönelik mesh akışlarını ve internet üzerindeki uçtan uca şifreli mesaj/arama yolunu kapsar. İnternet yolu güncel sürümlerde Signal protokolü (ileri gizlilik + kuantum-sonrası anahtar anlaşması) kullanır ve doğrulama için güvenlik numarası sunar. Public GATT kanalı hassas operasyon içeriği için uygun değildir.
Kurumsal koşullar	Uygulama dağıtımı, cihaz filosu doğrulaması, müdahale ekibi kimlik ve rol yönetimi, saha standart işletim prosedürleri ve acil servis eskalasyon notları önceden tanımlanmalıdır.

Başlık	Kısa açıklama
Kurumsal operasyon paneli	Crisis Connect Web 0.1.0; kurum panel yönetimi, saha ekibi telemetrisi, SOS sinyal gözlemleri, SSO/oturma yönetimi, lisans durumu, güncelleme bilgisi, denetim kayıtları ve kurulum health heartbeat takibi için ayrı bir operasyon kontrol düzlemi sağlar. Temmuz 2026 hattı buna uçtan uca şifreli mesajlaşma yüzeyi, MLS medya şifrelemeli SFU grup aramaları, Telegram/Slack asistan kanalları ve zamanlayıcı workflow otomasyon düzlemini ekler. Düz metin acil durum içeriği kontrol düzleminin dışında kalır; sunucular internet mesaj yolunda yalnızca şifreli zarfları yönlendirir.
Karar ölçütü	Bu sürüm, Android v1.1.7 referansını esas alır. iOS 1.1.2 istemcisi, iOS 17+ iPhone ve iPad cihazları için Android paritesi hedefiyle geliştirilen, CI/TestFlight hattına bağlı mağaza-listeli uygulama referansı olarak izlenmektedir. Kurumsal dağıtım kapsamı ise ancak kabul kapıları ve sahadan ölçülen kanıtlar karşılandıktan sonra genişletilmelidir; yeni internet düzlemi için saha kanıtı henüz çevrimdışı yol kadar derin değildir.

Contents

Document Purpose and Reader Guide	i
Executive Summary	ii
Kısa Kurumsal Özet (TR – Bilgilendirici)	v
Decision Context	1
System At A Glance	2
Deployment Model And Trust Boundaries	4
Operational Dashboard And Self-Hosted Control Plane	5
Security, Privacy, And Operating Constraints	6
Pilot Decision Framework	8
Recommended Institutional Rollout	10

Institutional Positioning	10
Technical Assurance Appendix	12
Document Governance	A-1
Terminology and Acronyms (Controlled Glossary)	A-3
1 Technical Context	A-6
2 Technical Scope and Non-Goals	A-6
3 Technical Design Goals and Principles	A-7
4 Technical Contributions	A-8
5 Technical System Overview	A-8
6 Reference User Workflow	A-10
7 Reference Communication Architecture	A-12
8 Hybrid Internet Transport Plane	A-24
9 Cryptographic Design	A-28
10 Rescue Mode	A-32
11 Voice Communication	A-34
12 Background Services	A-36
13 Cloud Services: Trust Establishment, Telemetry, and Ciphertext Relay	A-45
14 Web Dashboard and Self-Hosted Operations Architecture	A-48
15 Threat Model	A-84
16 Privacy Considerations	A-86
17 Evaluation Plan	A-89

18 Use Cases	A-107
19 Limitations and Safety Notes	A-108
20 Conclusion	A-109
A Implementation Status and Roadmap Notes	A-111

Decision Context

Disasters create a communications gap before formal infrastructure is restored. Crisis Connect is aimed at that gap: the window in which internet and cellular services are unavailable or overloaded, but nearby smartphones still have power and short-range radios.

This document is written as a decision artifact, not as a marketing brochure. The near-term institutional question is whether Crisis Connect is suitable for controlled pilot use as a local coordination layer for responder teams, pre-provisioned civilian groups, or mixed exercises in infrastructure-loss scenarios.

Decision dimension	Current decision-maker takeaway
Operational problem	Last-resort nearby coordination when wide-area communication fails first, with continuity across the surrounding partially-connected phases.
Primary transport	Local Bluetooth device-to-device transport is the default incident path. When validated internet connectivity exists, the same message queue additionally drains over an end-to-end encrypted internet relay; the offline path never depends on it.
Secure paths	QR-provisioned contact sessions, Rescue secure chat, role-gated responder mesh, and the internet E2E messaging/calling path are treated as designated secure channels.
Cloud dependency	The offline incident path has no cloud dependency. Optional cloud services support trust, telemetry, SOS-uplink, and push workflows, and on the internet path relay ciphertext-only message envelopes that backend infrastructure cannot decrypt.
Operations dashboard	Crisis Connect Web 0.1.0 supports authority panel operations, self-hosted deployment governance, SSO/session control, audit records, licensing, signed update visibility, installation-health reporting, an E2E messaging surface, SFU/MLS group calls, external assistant channels, and workflow automation; it is not part of the local incident message path.
Validated baseline	Current repository baseline under review is Android v1.1.7 (versionCode 57); the separate iOS client line is 1.1.2 for iOS 17+ on iPhone and iPad and is treated in this document as the store-listed parity implementation baseline, but it remains outside validated deployment scope.
Not positioned as	Replacement for national emergency-service infrastructure, certified public-safety radio systems, or unrestricted anonymous public mesh networking.
Board posture	Appropriate for evidence-driven authority pilots and staged public-launch preparation against the Android v1.1.7 baseline.

System At A Glance

Crisis Connect is intentionally conservative in how it expands communication scope. The default model is direct local exchange between known or proximate devices, with higher-

reach features introduced only under explicit policy and operator controls. Since the 1.1.x line, the same contact graph also operates over an internet-augmented end-to-end encrypted path when infrastructure survives; this hybrid model is offline-first by construction because the internet transport is an additional drain of the same local message queue, not a replacement for it.

Mode	Primary users	Security / trust posture	Intended use
Direct contact messaging	Pre-provisioned contacts	QR-first trust establishment with encrypted secure-path messaging.	Family, team, or roster-based local messaging when infrastructure is unavailable.
Internet E2E messaging and calls	Contacts with internet-capable identities	End-to-end encrypted envelopes (Signal-protocol sessions on current builds) relayed through ciphertext-only infrastructure; safety-number verification.	Reach beyond radio range while infrastructure survives; seamless continuity of the same conversation across transports.
Authority channel communications	Agency members and hierarchy peers	Shared agency keys issued by membership-gated backend; server holds ciphertext only; offline Bluetooth bridge fallback.	Agency-internal and cross-panel coordination interoperable with the web dashboard.
Rescue Mode	Victims and verified responders	Verified responder gate before secure-chat promotion; optional cloud uplink of active SOS observations when connectivity exists.	Proximity-based distress discovery and authenticated responder interaction.
Authorized responder mesh	Authorized responder roles only	Role-gated secure group chat over Wi-Fi Aware.	Sector-team coordination when compatible devices and responder credentials are available.
Public GATT general channel	Nearby participants who explicitly enable it	Non-confidential broad-coordination channel with bounded abuse controls.	Broad nearby coordination and general announcements when open local participation is needed.

What the system does not try to do. Crisis Connect does not replace cellular infrastructure, emergency call systems, or certified public-safety networks. It also does not claim unbounded relay/store-and-forward coverage or guaranteed reach through all obstruction conditions.

Deployment Model And Trust Boundaries

The deployment model is designed to keep plaintext emergency content on end-user devices while allowing institutions to retain governance over identity and responder authorization. Civilian participation assumes the app is already installed before connectivity loss or is distributed through a managed pre-incident workflow.

- **Offline incident content path.** Emergency text, Rescue secure chat, and responder mesh messages remain on local radio links between devices; this path has no cloud dependency.
- **Internet content path (ciphertext relay).** When validated connectivity exists, contact and authority-channel messages may additionally be relayed as end-to-end encrypted envelopes through backend infrastructure. The relay routes and stores ciphertext only, deletes relayed envelopes after acknowledgement, and applies a retention TTL to unacknowledged ones; plaintext never leaves the endpoints.
- **Trust establishment.** Institutions may operate certificate issuance, role governance, and related identity workflows on their own infrastructure.
- **Telemetry boundary.** Crisis Link synchronization and the SOS cloud uplink are operational metadata only and should be governed separately from incident content.
- **Public-channel policy.** The Public GATT channel is optional, disabled by default, and should be treated as a bounded coordination feature rather than a trusted confidential channel.

Boundary question	Institutional interpretation
Can the system work without the internet?	Yes for the local incident path; no cloud dependency is assumed for nearby device-to-device messaging, and internet-path failure degrades gracefully to Bluetooth-only operation.
What does the cloud see on the internet path?	Ciphertext envelopes plus routing metadata (sender/recipient identifiers, timing, push tokens). Content is end-to-end encrypted; relayed envelopes are deleted after acknowledgement with a TTL purge for the rest.
Where does responder trust come from?	Institution-controlled issuance and offline verification policy, anchored to a trusted authority key.
What requires prior preparation?	App distribution, QR onboarding for trusted civilian contacts, responder credential lifecycle, and pilot SOPs.
What remains sensitive even when encrypted?	Radio-visible metadata, internet-path routing/presence metadata, battery behavior, device compatibility, and open-channel misuse risk still require policy controls.

Operational Dashboard And Self-Hosted Control Plane

Crisis Connect Web **0.1.0** is the reviewed browser-based operations layer for institutional deployments. Its role is to give authorized operators a governed view over agencies, panels, users, responders, SOS signal observations, map markers, audit records, license state, update posture, and deployment health. The July 2026 line additionally provides an operator messaging surface (E2E-encrypted channels interoperable with the mobile clients), SFU group calls with MLS media encryption, external assistant channels, and a workflow automation plane. The web surface remains deliberately separated from the local incident message path: offline emergency content is exchanged device-to-device, while the dashboard consumes administrative state, operational metadata, and ciphertext-only messaging state.

Deployment boundary (restated). The dashboard should be described as an administrative, telemetry, governance, and operator-communication surface. It may read responder location snapshots, active SOS observations, panel metadata, audit logs, license status, update status, and health reports, and it participates in internet-path messaging strictly as an endpoint of end-to-end encrypted channels whose keys are issued to authorized members; backend collections hold ciphertext only. It must not be positioned as a relay for the offline local incident path, and institutional materials should keep this boundary explicit during procurement and pilot approval.

Current self-hosted maturity. The reviewed web workspace already contains a multi-stage Dockerfile using Node 20 Alpine, Next.js standalone output, a non-root runtime user, a `/api/health` healthcheck, runtime Drizzle migration artefacts, and Docker Com-

pose wiring for Postgres, Keycloak, MinIO, Redis, and the app. The Helm chart adds replica configuration, liveness/readiness probes, required secret injection, config/secret checksum restarts, a migration Job that runs before install and upgrade, optional license-renewal CronJob, and optional outbound health-report CronJob. This is a credible self-hosted control-plane baseline for evaluation and institutional hardening. Production deployments should still replace bundled evaluation data-plane components with operator-managed Postgres, Redis, MinIO/S3, and Keycloak services.

Known scope boundary. The main dashboard and several administrative APIs are on-prem aware, including user management, licensing, update status, audit writes, Crisis Sentinel provider/knowledge/model-release configuration, and workflow settings/execution endpoints. The deeper Settings surface, AI assistant/chat history, and parts of the workflow UI still contain Firebase client listeners or managed-service assumptions. Those areas should be migrated behind storage-backed server APIs before promising a fully Firebase-free administrative experience.

- **Authority visibility.** Server-guarded dashboard routes, role/agency checks, panel selection, responder/signal views, alert filters, and map markers give operations centers situational awareness without moving plaintext emergency chat payloads into any cloud relay.
- **Self-hosted deployment.** Cloud mode uses Firebase; on-prem mode uses Postgres/Keycloak through shared storage/auth abstractions, with Docker Compose for evaluation and Helm for Kubernetes promotion.
- **Lifecycle governance.** Enterprise OIDC, httpOnly sessions, ES256 licenses, signed update manifests, audit records, and optional installation-health reporting let authority IAM, release governance, and monitoring be reviewed separately from device-to-device emergency content.

Security, Privacy, And Operating Constraints

The security posture is strong where the product keeps scope narrow and explicit. Known-contact sessions use out-of-band trust bootstrap, Rescue secure chat is responder-gated, and the responder mesh requires role authorization. The weakest posture is intentionally the broadest feature: the Public GATT channel trades confidentiality for wider nearby coordination and therefore must be policy-limited.

Privacy remains a material concern because radio traffic can reveal presence and behavior even when payloads are encrypted. For institutional use, this means deployment approval should include advertisement hygiene, log anonymization, retention defaults, and clear operator guidance on when broad public coordination is acceptable.

Constraint area	Current posture	Board implication
Range and coverage	Strongly environment- and device-dependent.	Treat range as a pilot-measured operating envelope, not as a universal guarantee.
Voice communication	Optional and technically feasible, but evidence-sensitive.	Require scenario traces before relying on voice as an operational promise.
Battery and background behavior	Influenced by OEM power policy and OS constraints; hybrid operation adds push, presence, and connectivity-triggered work.	Validate on the target device fleet rather than assuming generic Android behavior.
iOS readiness	Store-listed iOS 1.1.2 parity implementation baseline with an active Codemagic CI/TestFlight pipeline, shared crypto golden vectors, and developer-recorded cross-platform interop checks.	Do not treat as institutional deployment baseline until representative device-matrix, BLE continuity, local-voice, background-policy, and interoperability-depth evidence close, and until the newest feature line passes through the release pipeline.
Internet-plane evidence depth	Internet E2E messaging/calling is source-reviewed with strong cryptographic design, but has no reviewer-grade field packet yet.	Gate pilot claims about internet-path delivery, push wake-up reliability, and TURN fallback on measured evidence, exactly as was required of the offline path.

Constraint area	Current posture	Board implication
Open-channel misuse	Managed by bounds, downgrade rules, and operator policy.	Do not frame Public GATT as an always-on trusted public instruction channel.

Pilot Decision Framework

The system should be judged on whether it can pass a controlled pilot with explicit evidence, not on architecture claims alone. The strongest current signal is source-conformance of the Android v1.1.7 baseline and its current role-gated/Public-GATT feature set. Battery evidence now includes a paired Pixel 7a idle-vs-standby trace set recorded on March 15, 2026, and voice evidence includes an adjacent-device baseline run, a short-range two-wall run, an initial packet-age instrumentation run, and a clock-adjusted latency run from March 15, 2026. These artifacts are useful pilot inputs, while broader field measurements and release-package QA remain part of the normal launch-readiness track.

Current readiness signal. As of July 17, 2026, the current repository baseline is Android v1.1.7 (versionCode 57), distributed through the public store workflow. One provenance caveat applies: the reviewed source line carries a small number of commits beyond the distributed 1.1.7(57) store artifact under the same version identifiers (the iOS VoIP calling push path, the reciprocal QR identity announce, and the platform-split message-push fan-out); a follow-up store build is expected to carry them, and reviewer packets should pin the exact commit. All three client repositories remain in active daily development, so this document's claims are bound to its revision date. The March 15, 2026 voice and standby artifacts and the March 31, 2026 review set remain attached as reviewer-visible supporting evidence for the offline path; they were captured on earlier builds and remain directional for the current line. The hybrid internet plane added on the 1.1.x releases is source-reviewed but has not yet produced a comparable field-evidence packet, and no offline-path acceptance gate is retroactively weakened by its addition.

Practical board reading. The document supports a controlled pilot decision if the institution is comfortable treating Crisis Connect as a bounded local-radio coordination layer with an internet-augmented E2E extension, tying formal go/no-go to the Android baseline, and carrying iOS 1.1.2 as the store-listed parity implementation baseline and parallel pilot workstream during phased launch steps.

Pilot gate	Suggested threshold	Current interpretation
Message delivery success	At least 95% indoor / urban; at least 85% rubble-like.	Gate is defined, but board confidence still depends on scenario evidence from field or controlled training runs.
Text latency	P95 less than or equal to 5 seconds.	Gate is defined; packet needs run-level latency evidence rather than only architectural rationale.
Voice envelope	Median one-way latency less than or equal to 150 ms and packet loss less than or equal to 3%.	March 15, 2026 adjacent-device and two-wall continuity runs plus an initial instrumentation run and a clock-adjusted debug run are now attached; the synchronized run reports 16–18 ms median and 99–120 ms P95 on the two instrumented devices, but broader degraded-condition scenario traces remain unclosed.
Power impact	Within mission-defined device limits.	A paired March 15, 2026 Pixel 7a idle-vs-standby trace set now exists, but it was captured on a debug, AC-powered build and broader release-equivalent comparison is still pending.
Responder workflow	100% scripted pass rate.	Appropriate as a hard gate because rescue trust failure is operationally unacceptable.
Public-channel abuse containment	99% malformed / out-of-window drop rate and zero trusted rendering of unverified announcements.	Governance model exists, but instrumented stress evidence remains pending closure.
Internet-path readiness (hybrid scope)	Relay delivery at least 99% within 10 s with 100% queue recovery; push wake-up at least 95%; 100% transport-handover continuity.	Gates are newly defined for the 1.1.x hybrid plane (G-06 / G-07); the plane is source-reviewed with cross-platform test vectors but has no field-evidence packet yet.

Recommended Institutional Rollout

An institutional rollout posture should be phased, evidence-gated, and explicit about scope.

1. **Closed responder pilot.** Validate device fleet behavior, credential issuance, Rescue Mode workflow, and message-path evidence using trained personnel only — including, where hybrid features are in scope, internet-path delivery, push wake-up, and transport-handover checks against the new evidence gates.
2. **Mixed controlled exercise.** Add bounded civilian or volunteer participants who are pre-provisioned and briefed on non-goals, public-channel policy, and escalation to official emergency services.
3. **Conditional broader deployment.** Expand only after voice, battery, privacy, abuse-governance, and (for hybrid scope) internet-plane evidence are attached to the institutional packet and reviewed against mission thresholds.

This rollout sequence matches the system’s strengths: explicit trust boundaries, offline-first transport with an evidence-gated internet extension, and measurable pilot gates. It also avoids overstating the product in the exact areas where field conditions can invalidate desk-level assumptions.

Institutional Positioning

Positioning area	Current statement
Primary fit	Off-grid coordination, earthquake response, and proximity-based search support where nearby devices can still communicate locally, with end-to-end encrypted continuity over surviving infrastructure before, during, and after the incident window.
Best-fit operator model	Public authorities, trained responder teams, pre-provisioned organizations, and controlled mixed-user exercises.
Not positioned as	A universal public emergency network, a replacement for certified emergency-service infrastructure, or an unmanaged anonymous relay platform.
Go/No-Go basis	Delivery, latency, voice stability, power impact, responder workflow success, abuse-governance evidence, internet-plane readiness evidence where hybrid features are in scope, and deployment governance readiness.

Decision statement. Crisis Connect should be presented as an evidence-gated local emergency communication layer for controlled institutional adoption and staged public-launch preparation. It should not be presented as a generalized nationwide emergency-network claim.

NEXT SECTION

Technical Assurance Appendix

The authority brief ends here. The next part contains the technical context, protocol details, governance language, cryptographic profile, validation framework, and reviewer evidence materials that support institutional assessment.

PART II

Technical Assurance Appendix

This appendix contains the detailed engineering basis, governance language, protocol profiles, cryptographic assumptions, privacy controls, evaluation framework, and reviewer materials that support the authority brief. It is intended for technical reviewers, security assessors, institutional pilot teams, and formal authority assurance workflows.

Appendix Navigation

Appendix element	How to use it
Document governance	Version control, terminology rules, audience definition, and revision record for the issued whitepaper.
Protocol and trust model	Detailed transport profiles (offline Bluetooth/mesh and the hybrid internet plane), trust assumptions, cryptographic controls, and governance constraints behind the public whitepaper claims.
Web operations control plane	Dashboard architecture, deployment modes, data model, SSO/session model, Docker/Helm packaging, licensing, update, and health-reporting controls for <code>crisis-connect-web</code> .
Validation and evidence	Acceptance gates, risk register, QA/evidence snapshots, and reviewer-visible supporting artifacts.
Reviewer packet material	Release-package checklist, reviewer access references, crypto profile sheet, board card, and traceability matrix.
Final controls	Disclaimer, limitation-of-liability language, and publication-governance closing notes.

Document Governance

Purpose. This whitepaper is an institutional decision artifact for pilot Go/No-Go evaluation, not a public marketing document.

Normative language. The capitalized terms **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** indicate requirement strength for implementation, governance, and pilot acceptance decisions.

Evidence discipline. Architecture claims are treated as provisional until matched with pilot evidence in the Evaluation Plan section and tracked through claim-level review artifacts.

Language standardization. The normative document language is English. A secondary-language executive summary **MAY** be provided for stakeholder accessibility, but it is strictly informative and **MUST NOT** redefine requirements, threat assumptions, or acceptance criteria.

Terminology control. The section “Terminology and Acronyms (Controlled Glossary)” is the authoritative vocabulary for this document. Normative sections **MUST** use glossary-defined terms consistently; synonyms or local aliases **SHOULD** be avoided unless explicitly mapped once to the glossary term.

Reference quality policy. References MUST include a DOI where the publisher assigns one. Online references MUST include a stable URL and an access date (`urldate`). If no DOI exists, the entry MUST explicitly state that DOI is not assigned.

Change control. Any change that affects transport guarantees, trust boundaries, cryptographic behavior, or acceptance gates MUST update the document version, revision history, and claim-evidence mapping.

Edition record. Controlled publication metadata and revision lineage for this issued edition are recorded below.

Document Version	2.0.0
Revision Date	July 17, 2026
Status	Authority Review Edition
Document Structure	Decision-maker Whitepaper + Technical Assurance Appendix
Document Owner	Auralis Industries
Intended Audience	Institutional review boards, emergency response authorities, and technical security reviewers
Classification	Public (pilot evaluation)
Distribution	Emergency authorities / launch stakeholders
Validated Android Baseline	Crisis Connect Android v1.1.7 (versionCode 57) source baseline
Reviewed Web Baseline	Crisis Connect Web 0.1.0 local dashboard / on-prem operations baseline

Version	Date	Change Summary
2.0.0	July 17, 2026	Major hybrid-architecture revision: baselines re-anchored to Android v1.1.7 (versionCode 57) and iOS 1.1.2. Documents the internet E2E messaging plane (ECIES envelope v2, Signal-protocol v3 with PQXDH + Double Ratchet), anonymous internet identity and default-on registration boundary, SPAKE2 add-by-number bootstrap, internet WebRTC calls and cross-platform GATT calls, SFU/MLS group calls, authority channel communications with Bluetooth bridge, SOS cloud uplink and contact escalation, Crisis Sentinel on-device/online AI, mesh accelerators and PTT, 19-language localization, iOS CI/TestFlight posture, and the July 2026 web control-plane expansion (messaging/SFU, Telegram/Slack assistant channels, workflow scheduler/MCP, hardened Firestore rules). Trust-boundary, threat-model, privacy, and claim-evidence language rewritten for the hybrid model.
v1.1.0	June 6, 2026	Web operations update: added Crisis Connect Web 0.1.0 dashboard review, Settings/AI/workflow control-plane analysis, self-hosted Docker/Helm architecture, Postgres/Keycloak on-prem path, licensing/update/health governance, and dashboard-specific trust-boundary language while retaining Android v1.0.0 as the institutional deployment baseline.
v1.0.0	April 5, 2026	Android v1.0.0 alignment refresh: whitepaper version normalized to 1.0.0, current source provenance anchored to the public source repository, and reviewer-facing release language simplified for the 1.0.0 launch line.
v0.9.0	February 19, 2026	Initial institutional whitepaper baseline (pre-release draft lineage consolidated before external publication).

Terminology and Acronyms (Controlled Glossary)

Term	Meaning in this document
BLE	Bluetooth Low Energy; used for discovery, control, and Rescue Mode signaling.
RFCOMM	Bluetooth Classic serial transport profile used for point-to-point payload transfer.
Wi-Fi Aware	Android proximity networking capability used for role-gated responder mesh discovery and secure group-chat sessions in the current v1.1.7 scope.
Public GATT mesh	Opt-in BLE GATT-based nearby general-chat channel (multi-device relay with bounded hop policy) used for broad local coordination in v1.1.7.
GATT	BLE data model for services/characteristics used in SOS responder handshake flows.

Term	Meaning in this document
PSK	Pre-shared key provisioned out-of-band (QR) and used for contact secure-path activation.
ECDH	Elliptic-curve Diffie-Hellman used for ephemeral key establishment in responder sessions.
AEAD	Authenticated encryption with associated data; secure-path baseline profiles use AES-GCM for confidentiality and integrity.
IAM	Identity and access management workflow used to issue responder credentials.
Crisis Link	Optional operational telemetry synchronization channel for authorized responders; does not carry emergency message content.
Crisis Connect Web	Browser-based operations dashboard and self-hosted control plane for authority panels, users, responder telemetry, SOS observations, licensing, updates, audit logs, and installation health; since July 2026 also an endpoint of end-to-end encrypted operator messaging and SFU/MLS group calls (backend collections hold ciphertext only); not a relay for the offline local incident path.
On-prem mode	Self-hosted web deployment mode using Postgres, Keycloak/OIDC, MinIO/S3-compatible storage, Redis, Docker Compose or Kubernetes/Helm.
CRL / denylist snapshot	Revocation artifact distributed during connectivity windows for offline verification (governance target; parser/enforcement is not part of the validated v1.1.7 app baseline).
Internet E2E relay	Backend message path (Firebase Cloud Functions/Firestore) that routes end-to-end encrypted envelopes between internet-capable contacts; it stores ciphertext only, deletes envelopes after recipient acknowledgement, and purges unacknowledged envelopes on a TTL.
ECIES envelope (v2)	Authenticated sealed-sender message profile: P-256 ECDH (ephemeral-static + static-static) with HKDF-SHA-256 key derivation and AES-256-GCM, with sender/recipient/conversation identifiers bound as AAD.
Signal v3 session	Forward-secret internet messaging profile using the Signal protocol: PQXDH (post-quantum, Kyber-based) key agreement plus Double Ratchet, with TOFU identity pinning, anti-downgrade rules, and numeric-fingerprint safety numbers.
PQXDH	Post-quantum extended Diffie-Hellman: Signal's initial key-agreement protocol combining elliptic-curve exchanges with a Kyber post-quantum KEM, protecting session setup against future quantum adversaries.

Term	Meaning in this document
Double Ratchet	Signal's per-message key-evolution algorithm providing forward secrecy and post-compromise healing within an established session.
Prekey	Key material published to the directory in advance (signed, one-time, and last-resort Kyber prekeys) so a peer can start a Signal session while the recipient is offline; pools are topped up automatically and one-time prekeys are consumed on use.
TOFU	Trust-on-first-use: the first observed peer identity key is pinned; any later change raises a user-facing key-change warning that requires out-of-band re-verification before the conversation is trusted again.
Safety number	Human-comparable numeric fingerprint of both parties' identity keys (libsignal format) used to verify an internet-path contact out of band.
SPAKE2	Password-authenticated key exchange (RFC 9382, P-256 profile) used for harvest-resistant add-by-phone-number contact bootstrap over a number-free BLE beacon.
SFU / MLS	Selective-forwarding group-call architecture (Cloudflare Realtime) whose media frames are end-to-end encrypted with a Messaging Layer Security group key; the SFU forwards but cannot decrypt frames.
WebRTC / ICE / STUN	Standard real-time media stack used for internet calls: WebRTC carries encrypted voice/video, ICE negotiates the network path, and STUN discovers direct peer routes; TURN (separate entry) relays media only when direct paths fail.
TURN	Relay servers used by WebRTC calls when direct peer paths fail; per-session credentials are minted server-side and media remains encrypted end-to-end above the relay.
FCM / APNs	Platform push services (Firebase Cloud Messaging on Android; Apple Push Notification service on iOS) used for silent data-only wake-ups; the server holds no plaintext to display, so clients decrypt locally and render their own notifications.
PushKit / CallKit	Apple frameworks for VoIP push wake-up (including from force-quit state) and native incoming-call presentation; the Android counterpart is the Telecom framework.
Play Integrity / App Check / App Attest	Platform attestation services (Google and Apple) used to verify that backend callables are invoked by genuine, untampered app builds before relay operations are served.

Term	Meaning in this document
MCP	Model Context Protocol: a standardized tool-invocation interface for AI agents. In this document, a bearer-authenticated JSON-RPC endpoint through which approved external agents may invoke published crisis workflows under panel-scoped rate limits and audit logging.
RAG	Retrieval-augmented generation: grounding AI answers in panel-scoped, authority-approved knowledge (runbooks, incident precedent) retrieved from a vector store, with citation-tagged context.
Crisis Sentinel (mobile)	On-device AI assistant runtime (LiteRT) with managed model downloads plus an authenticated online engine backed by the web control plane; decision support, not a message transport.
Mesh PTT	Push-to-talk voice over the authority mesh with single-speaker floor control and multi-hop-aware participant addressing.
Pilot gate	Institutional acceptance checkpoint based on predefined KPI thresholds and required evidence.

1 Technical Context

Disasters frequently disrupt communication infrastructure (cell towers, power, backhaul). Crisis Connect targets the last-resort proximity regime where nearby devices can still communicate even when wide-area networks are unavailable. The design goal is not to replace official emergency services, but to provide a resilient local communication layer to support coordination and victim discovery.

2 Technical Scope and Non-Goals

2.1 Scope

- Local, offline device-to-device messaging and coordination over Bluetooth.
- End-to-end encrypted internet messaging between contacts as a third transport of the same message queue, relayed through ciphertext-only backend infrastructure (v1.1.7 scope; Signal-protocol sessions on current builds).
- End-to-end encrypted internet voice / video calls (WebRTC) and offline Bluetooth GATT voice calls, including cross-platform Android–iOS calls.
- Authority (agency) channel and hierarchy messaging interoperable with the web control plane, with an offline Bluetooth bridge fallback.

- Role-gated responder secure group chat over Wi-Fi Aware (v1.1.7 extension scope), with optional Wi-Fi Aware / Wi-Fi Direct media accelerators and mesh push-to-talk.
- Opt-in Public GATT mesh general-chat channel for broad nearby coordination (v1.1.7).
- Secure contact communication via pre-shared keys provisioned out-of-band (QR) or via SPAKE2 add-by-number bootstrap.
- Authenticated responder interaction for Rescue Mode operations, with optional SOS cloud uplink and automatic contact escalation when connectivity exists.
- Optional Crisis Link telemetry synchronization for authorized responders (out of emergency message content path).
- Crisis Sentinel decision-support AI: on-device runtime with managed model downloads plus an authenticated online engine.
- Optional Crisis Connect Web dashboard for authority operations, operator messaging/group calls, self-hosted identity/storage, deployment licensing, update visibility, and installation health oversight.

2.2 Non-goals (baseline design)

- Providing wide-area connectivity or replacing cellular networks; the internet path uses existing infrastructure and disappears with it.
- Unbounded anonymous relay/store-and-forward networking without explicit operator opt-in, policy controls, and bounded runtime limits.
- Guaranteeing delivery beyond Bluetooth proximity constraints when infrastructure is down.
- Exposing plaintext emergency content to the web dashboard, relay backend, or any cloud service; internet-path relay is ciphertext-only by design.

3 Technical Design Goals and Principles

- Correctness under stress: minimize state and failure modes in chaotic environments.
- Offline-first operation: the local incident transport has no infrastructure dependency; internet augmentation is additive and its failure must never degrade the offline path.
- End-to-end confidentiality everywhere: any path that leaves local radio range carries ciphertext only, with endpoint-held keys.
- Energy awareness: support standby readiness and bounded background work.

- Verifiable trust: clear trust boundaries, auditable authentication paths, and user-verifiable safety numbers on the internet path.

4 Technical Contributions

- A practical offline transport model where Bluetooth remains the default local channel and authorized responder teams may activate a controlled Wi-Fi Aware secure group-chat extension.
- A hybrid store-and-forward design in which one message queue drains over RF-COMM, BLE/GATT, and an internet E2E relay, so a single conversation survives transport changes without user action.
- A cross-platform end-to-end encryption profile (ECIES envelope and Signal-protocol sessions) kept byte-compatible across Android, iOS, and web through shared golden test vectors.
- A two-stage trust model combining pre-shared secrets for known contacts with authenticated rescue interactions for verified responders, extended with SPAKE2 add-by-number bootstrap and safety-number verification for internet identities.
- An evaluation plan suitable for institutional trials, including metrics, test settings, and reporting requirements.
- A deployable operations dashboard baseline with cloud and self-hosted modes, allowing institutional reviewers to separate plaintext incident transport from identity, telemetry, audit, licensing, health governance, and ciphertext-only messaging state.

5 Technical System Overview

5.1 Operating assumptions

- The app must function when internet and cellular networks are unavailable or overloaded; connectivity, when present, is treated as an opportunistic bonus rather than a dependency.
- Users may have limited time and power budget; background readiness is required.
- Civilian participation assumes that a compatible app build is already installed on the device before connectivity loss, or is distributed through an approved local workflow while installation remains possible.
- Offline communication is proximity-based: peers must be within local radio range (Bluetooth by default; Wi-Fi Aware for authorized responder mesh sessions). Internet-

path communication requires validated connectivity on both endpoints at some point, with store-and-forward covering the gaps.

5.2 Core capabilities (current scope)

- Secure text messaging and location updates over Bluetooth.
- End-to-end encrypted internet messaging for the same contacts, including low-bandwidth template messages, chunked encrypted attachments, delivery/read receipts, and push-based delivery; a conversation started over Bluetooth continues over the internet using the same conversation identifier.
- Voice communication on three paths: point-to-point Bluetooth RFCOMM (Android–Android), encrypted Bluetooth GATT calls (cross-platform, including Android–iOS), and internet WebRTC calls with video and screen share (codec: Opus; see the Voice Communication and Internet Calls sections).
- Rescue Mode for SOS broadcasting and responder interaction, plus optional SOS cloud uplink to authority dashboards and automatic SOS alerts to top internet-capable contacts.
- Authority channel and hierarchy messaging for agency users, interoperable with the web dashboard, with encrypted attachments/voice notes and an offline Bluetooth bridge.
- Authorized responder secure group chat over Wi-Fi Aware mesh sessions, with optional media accelerators and mesh push-to-talk.
- Crisis Sentinel on-device/online AI decision support: the on-device engine applies an output validator and safety-coverage layer over model responses before display and includes an offline survival-guide knowledge index with citations, role-aware incident extraction, and speech dictation; the online engine is authenticated and role-gated (field-team and above) but does not require the on-device model download or a rescue device certificate — an access condition institutional AI-governance packets should record.
- Recent Disasters situational feed: a read-only informational surface aggregating public earthquake/disaster sources (GDACS, AFAD, Kandilli, USGS, EMSC) with region-aware source selection, fetched over the internet when available.
- Optional Crisis Link telemetry synchronization with offline queueing.
- Optional Crisis Connect Web dashboard for agency panels, responder telemetry, SOS signal visualization, server-side session checks, license status, update status, and installation health reporting.
- On-demand responder packaging through the dynamic `:feature_rescue` module for Rescue UI/services in the current Android project layout.

5.3 Information hygiene and rumor containment

A recurring failure mode in disasters is the rapid spread of unverified information. Crisis Connect therefore distinguishes between:

- Contact-to-contact messaging: by default, civilians exchange messages only with pre-provisioned contacts (paired out-of-band), limiting unsolicited broadcasting and reducing accidental rumor propagation.
- SOS broadcast: victims may advertise an SOS beacon without prior pairing; however, two-way SOS communication is restricted to verified responders, limiting abuse while enabling rapid assistance.

Abuse/DoS note (Public GATT channel). The public channel is intentionally broad-reach and non-confidential, so misinformation and spam pressure are expected failure modes. Pilot posture **MUST** enforce bounded forwarding and ingestion controls (hop cap, timestamp-validity window, inbound packet budget, and message-size caps), **SHOULD** provide user-level mute/report workflow coverage, and **SHOULD** reserve a responder-only announcement format for official instructions.

6 Reference User Workflow

6.1 Pre-incident: QR-based contact provisioning

Trusted contacts are provisioned before an incident by exchanging QR codes. The QR exchange is an out-of-band channel used to establish shared secrets and reduce in-band key bootstrap requirements during first contact.

Operational constraint. QR provisioning is practical for a bounded trust set (e.g., family, a small response team, or an organization-defined roster).

Large-scale, ad-hoc population-wide pairing is not assumed; institutions should plan onboarding workflows (training, pre-registration, or managed device fleets) consistent with their operational model.

Deployment prerequisite. Civilian users can participate only on devices that already have a compatible application build installed. If broader public use is a deployment goal, authorities should define a pre-incident distribution and installation workflow rather than assuming first-time download will be available after connectivity loss.

6.2 QR payload profile and onboarding safeguards (implementation baseline)

In the current Android v1.1.7 baseline, QR onboarding accepts multiple payload styles to reduce field failure:

- DCS URI + JSON: `dcs://{...}` with canonical fields for session code and shared key.
- URI-encoded JSON variants (decoded with bounded retry).
- Legacy pipe-delimited payloads for backward compatibility.

Discovery policy is conservative and explicit:

- Primary device-name discovery timeout: 15 seconds.
- Secondary fallback timeout for alternative candidate names: 8 seconds.
- Post-pairing activation gate: encrypted challenge-response handshake over RFCOMM with a 10-second timeout before contact activation.

This profile keeps the trust decision out-of-band (QR) while preventing contacts from being marked ready until cryptographic liveness is confirmed.

6.3 Incident: direct offline communication

During an incident, devices discover each other locally and establish a direct connection. The offline message path remains local and infrastructure-free. If a contact's Bluetooth link is not currently up and validated internet connectivity exists, the same conversation fails over to the end-to-end encrypted internet relay described in the Hybrid Internet Transport Plane section; loss of connectivity returns the conversation to Bluetooth-only operation without user action.

6.4 Contact bootstrap beyond QR (implementation baseline)

Two additions extend the QR-first trust model on the current baselines:

- **Unified QR contacts.** QR pairing now provisions both a Bluetooth key and, where available, the peer's internet messaging identity in one flow. Identity exchange is bidirectional: the scanner's identity is carried in the pairing client-hello with an HMAC proof bound to the session nonces, and an MTU-independent fallback announce over the E2E relay heals one-sided reachability. If Bluetooth verification fails in the field, the flow can fall back to an internet-only contact so the pairing intent is not lost.
- **SPAKE2 add-by-number.** Nearby contacts can be added by phone number without broadcasting anything derived from that number. The advertiser emits a generic, number-free "open to add" BLE beacon; the seeker runs a SPAKE2 (RFC 9382, P-256) handshake using the phone number as the password.[1] A wrong guess fails at key confirmation and reveals nothing (one guess per live handshake). On success, peers exchange messaging identities encrypted under the SPAKE2-derived key — behind

an explicit accept/decline consent gate for unknown peers (a handshake whose exchanged identity exactly matches an already-saved contact, same account and same identity key, is accepted without a prompt, since it only re-establishes a Bluetooth route for an existing trust decision) — and create a unified contact with both an internet identity and a Bluetooth key. The feature is off by default and auto-enables only after the user adds a phone number, never overriding an explicit opt-out. This design replaced an earlier truncated-hash discovery beacon precisely because hash-based beacons are vulnerable to offline dictionary harvesting.

7 Reference Communication Architecture

7.1 Primary Bluetooth design (baseline default)

Crisis Connect uses Bluetooth Low Energy (BLE) for discovery and short control exchanges, and Bluetooth Classic (RFCOMM/SPP) for higher-throughput point-to-point transport.[2, 3]

7.2 Bluetooth layers

BLE (GAP/GATT): low-power discovery and presence broadcasting; authentication handshakes; SOS beacon behavior in Rescue Mode.[2]

Bluetooth Classic (RFCOMM/SPP): reliable point-to-point transport for text messaging and optional voice streaming.[3]

Bluetooth security assumptions (explicit)..

- Contact activation is pairing-led in onboarding flows: a contact is not marked operational until OS-level pairing/bonding and post-pairing encrypted challenge-response succeed.
- RFCOMM compatibility posture includes both secure and insecure socket paths (listener and connector fallbacks) to tolerate mixed device stacks during incidents.
- Therefore, transport confidentiality/integrity **MUST** be anchored at the application layer (challenge-response + AEAD on designated secure paths), not by socket mode alone.
- High-assurance deployments **SHOULD** disable insecure RFCOMM fallbacks by policy/profile and treat unsuccessful pairing/handshake completion as a hard no-send state.

7.3 Authorized Wi-Fi Aware mesh extension (v1.1.7 responder scope)

Android baseline v1.1.7 includes an optional, role-gated secure group-chat extension over Wi-Fi Aware for authorized responder roles (`admin/fieldteam`). This extension remains separate from the Public GATT mesh channel, is packaged via `:feature_rescue`, and is disabled unless the device satisfies both authorization and platform-capability checks.

- Control plane: `MeshAwareService` foreground runtime with Wi-Fi Aware session management and bounded reconnect backoff.
- Authorization gate: local role-certificate validation is required before mesh activation; unauthorized join attempts are rejected and surfaced as operator alerts.
- Session security: peers exchange authenticated proofs and derive per-peer session keys using ECDH P-256 + HKDF-SHA-256; payloads use AES-256-GCM with message-bound AAD.
- Operational scope: secure group chat requires at least two verified devices and remains local-radio only.

From an operational-readiness perspective, this feature is intentionally not the default public posture. Android requires both device support and current availability checks for Wi-Fi Aware, and active Aware sessions keep the device synchronized to an Aware cluster until closed, which consumes resources and battery.[4, 5] This makes the feature a better fit for authorized, time-bounded responder coordination than for universal civilian standby.

7.4 Opt-in Public GATT mesh channel (v1.1.7)

In the v1.1.7 baseline, a separate Public GATT mesh channel can be enabled from Advanced Settings (“Public mesh mode”) for broad nearby coordination and general announcements. This channel is intentionally distinct from role-gated responder mesh controls.

- Activation policy: disabled by default; started only when a user explicitly enables “Advanced Settings > Public mesh mode” (stored preference key: `advanced_public_mesh_enabled`).
- Access behavior: users who do not enable this setting cannot use general mesh chat; UI keeps general chat disabled and prompts enabling Public mesh mode from Advanced Settings.
- Transport/runtime model: BLE GATT foreground service with scanning/advertising and bounded queue/retry logic for local-radio propagation.
- Participation posture: nearby active devices can join the general channel; message propagation may traverse multiple devices under bounded hop controls.

- Security boundary: this channel is designed for broad coordination, not confidential responder traffic. In the current v1.1.7 baseline, Public GATT chat packets can carry an AES-GCM envelope (`keyId/iv/cipher` + AAD-bound metadata) under a shared public-channel key profile; this is not role-gated confidential E2E protection. Sensitive operational content should use role-gated secure responder channels.
- Operational recommendation: this mode is not recommended as always-on default for all users; it SHOULD be enabled only when nearby broad-coordination need is explicit.

From a disaster-operations perspective, this channel SHOULD be treated as a situational coordination overlay rather than as the default earthquake/collapse posture for all users. Public relay mode adds ongoing scan/advertise/queue work; Android reserves highest-duty-cycle BLE scanning for foreground use and enforces low-power scanning outside the foreground, while smartphone BLE measurements show scanner energy is higher than advertiser energy and rises with larger scan windows.[6, 7] In practice, Public GATT mesh is sensible for bounded, time-limited nearby coordination windows, but not as an always-on default for stressed civilian populations.

Declared operational parameters (Public GATT pilot lock).. These values MUST be declared unchanged in pilot evidence packets (or explicitly exception-approved) so reviewers can evaluate rumor containment and abuse resistance against a stable operating profile.

Responder-only announcement format (pilot governance profile).. To reduce rumor spread on Public GATT, authorities SHOULD reserve high-priority incident announcements for a verifier-gated payload profile. This is a pilot governance contract (not a claim that all peers natively parse it today) and MAY be carried as structured JSON or canonical text as long as the signed tuple is preserved. In the validated v1.1.7 source baseline, an explicit parser token for `responder_announcement_v1` is not yet observed in Public GATT runtime; trusted-announcement rendering therefore remains an evidence-closure item under C-06/T-08.

Validation policy for pilots:

- Signature, certificate chain/trust anchor, role authorization, and expiry checks MUST succeed before rendering as trusted announcement.
- Invalid signature, wrong role, or expired payload MUST be downgraded to untrusted public chat representation.
- Unverified announcement payloads MUST NOT trigger priority siren/banner UX reserved for trusted responder instructions.

Control parameter	Declared value (v1.1.7 baseline)	Drop / enforcement behavior
Forward-hop ceiling	MAX_FORWARD_HOPS = 4.	Packets at/above hop ceiling are accepted locally but not relayed further.
Hop-field validity	Accepted hop value range: 0..4.	Packets outside range are rejected before processing.
Timestamp acceptance window (runtime)	Packet timestamp MUST be within now-24h and now+2m.	Packets outside this window are dropped as invalid before local accept/relay.
Operational freshness policy (pilot)	Public chat older than 10 minutes MUST be marked STALE; responder announcement older than 30 minutes MUST be marked STALE-URGENT.	Freshness labels are mandatory for operator interpretation; packet acceptance still follows runtime timestamp window.
Inbound flood budget	180 packets per source address per 10 s window.	Over-budget packets are dropped for that window.
Message/body size limits	Chat message max 1024 chars; packet text max 4096 chars; transport packet max 2048 B payload (+2 B header).	Oversized payloads are rejected before decode/relay.
Payload envelope profile	Chat payload MAY be emitted with AES-GCM envelope fields (keyId, iv, cipher); current channel profile uses key ID public-v1.	Envelope hardening improves integrity/format robustness but policy posture remains non-confidential public coordination.
Duplicate and replay horizon	Message-ID cache up to 4096 IDs, aged out at 24h.	Duplicate IDs are dropped; expired IDs are cleaned from cache.
Queued relay backlog cap	Pending outbound queue max 200 packets.	Oldest queued packets are evicted under sustained pressure.

7.5 Range envelope and environment dependence

Bluetooth range is highly environment- and device-dependent, and RSSI-only distance inference becomes unstable under multipath and obstruction.[8, 9] To avoid over-claiming, we define an expected operating envelope and explicitly distinguish best-case from typical conditions. All values below should be treated as preliminary targets pending controlled measurement.

Field	Declared rule
type	MUST be responder_announcement_v1.
issuerRole	MUST represent an authorized responder role (<code>admin/fieldteam</code>) validated against role certificate scope.
issuedAtMillis, expiresAtMillis	MUST be present; <code>expiresAtMillis</code> SHOULD be within 30 minutes of issuance for crisis instructions.
incidentScope	SHOULD include sector/zone identifier so recipients can assess relevance.
message	SHOULD be concise operational instruction text suitable for broadcast rendering.
issuerCertRef + sig	MUST bind issuer identity and ECDSA P-256 signature over canonical tuple (<code>type issuedAt expiresAt scope message issuerCertRef</code>).
Scenario	Indicative range (order-of-magnitude)
Open area / line-of-sight	50–150 m typical; best-case up to a few hundred meters on compatible hardware.
Urban / mixed obstacles	30–80 m.
Indoor (walls, corridors)	10–30 m.
Rubble / heavy obstruction	5–20 m (high variance).

The rubble / heavy-obstruction row is a conservative operational prior informed by smartphone-based victim-finding studies in disaster environments.[10, 11] Those studies report high variance from attenuation and device orientation effects.

7.6 Operational adequacy for collapse scenarios

Range adequacy is defined against mission context, not theoretical maximum distance. In earthquake and building-collapse operations, Crisis Connect is intended for close-proximity sector sweeps where responders and victims intermittently enter direct Bluetooth range.

Accordingly, the range envelope in this section is treated as an operational design target rather than a universal guarantee for all victims under all debris conditions. Deployment approval should be based on pilot evidence against the acceptance gates in the Evaluation Plan, including rubble-like scenarios.

If a peer is temporarily out of range, delivery is deferred until direct proximity is re-established; local retry/queue behavior depends on implementation policy.

7.7 Messaging protocol (high level)

Secure paths (contact RFCOMM sessions, Rescue secure chat, role-gated Wi-Fi Aware responder mesh) encrypt messages on the sender device and decrypt on the receiver device. The opt-in Public GATT general channel in v1.1.7 can carry AES-GCM-wrapped chat payload fields, but because it operates with a shared public-channel profile and broad participation posture, it **MUST** still be treated as non-confidential coordination traffic for governance decisions. If a peer is not currently connected, delivery is deferred until a link is available; local queuing is implementation-dependent.

7.8 Messaging protocol (wire format and safeguards)

Protocol profile (implemented baseline). In Android baseline v1.1.7, RFCOMM application traffic is newline-delimited UTF-8 records with two frame families: pipe-delimited control/data lines and JSON records for rich media and call-control sessions.

Pipe-delimited frame family (baseline).

Frame	Purpose and safeguards
HSK_REQ challenge ivB64 cipherB64	Challenge-response request. Challenge payload is AES-GCM protected. Production baseline carries no key material in this frame; legacy trailing fields are parse-tolerated for compatibility but ignored for key selection.
HSK_ACK challenge ivB64 cipherB64	Returns encrypted RESPONSE:challenge; handshake timeout is 10 seconds in baseline.
CONTACT_INFO ivB64 cipherB64	Encrypted contact metadata synchronization: session code, display name, avatar payload, and optional key echo under established secure context.
SEC_MSG messageUuid ivB64 cipherB64	Secure text message path; payload decrypted with contact AES key.
ACK messageUuid	Delivery/read progression marker on RFCOMM text path.
MSG ... and legacy prefixes	Backward-compatibility fallback; retained for interoperability during transition.

JSON frame family (baseline). Voice/image/file transfer and call-control flows use typed JSON records (voice:*, image:*, file:*, call:*; see Sections 7.10–7.11).

Nonce, replay, and idempotence safeguards. For AES-GCM, nonces must be unique per key.[12] The baseline profile uses 12-byte IVs generated from a CSPRNG per encrypted frame with a 16-byte authentication tag. To reduce operational collision risk and replay ambiguity, pilot deployments **MUST** enforce (i) bounded encrypted-frame volume per session key, (ii) key refresh on restart or re-handshake, and (iii) receiver-side replay cache

checks over sender/message UUID tuples with bounded TTL. Message-level idempotence relies on UUID-based persistence uniqueness; duplicate UUIDs are dropped without state advancement. As of this revision, local unit tests directly cover AES-GCM round-trip and certificate-shape checks; dedicated automated suites for ECDH/HKDF vectors and replay/rate-limit negative paths are still a pending expansion item.

Ordering semantics. Transport delivery is best-effort and does not guarantee strict global ordering. Where session sequence metadata is available, receivers SHOULD prefer sequence-aware rendering; otherwise arrival order is used while duplicate UUIDs remain suppressed.

Session establishment and key negotiation. Crisis Connect has two trust settings with different operational needs:

- Known contacts (PSK): key material is provisioned primarily out-of-band (QR) and used for handshake-based activation.
- Operational behavior (Android baseline): after successful handshake, encrypted contact metadata may synchronize profile state while trust bootstrap remains QR-first.
- Responder sessions: responder identity is verified via role proof over BLE; after ECDH key agreement, peers use a fresh session key for secure chat.

Protocol hardening note. `HSK_REQ|challenge|ivB64|cipherB64` no longer carries key-carry fields in the production baseline. Parser compatibility for legacy trailing fields is retained only to avoid hard failures with older peers; such fields are ignored for trust/key decisions.

SOS mode over BLE GATT (implementation-derived). In SOS mode, the victim device advertises a connectable BLE service and exposes a small set of GATT characteristics to support authenticated interaction with responders. On Android, this is implemented as a foreground GATT server that:

- advertises a connectable service UUID derived from assigned number 0xCC00 (current baseline advertisement payload carries service UUID only),
- derives an ephemeral session key via ECDH after receiving a client public key on an `auth_challenge` write, and returns the server public key via `auth_response`,
- requires a responder role proof before accepting chat traffic (role proof is delivered over a chunked write path),
- exchanges encrypted chat messages via a pair of characteristics (secure chat in/out) and uses a dedicated ACK characteristic.

This standby posture is intentionally lighter than public relay modes: the victim device mainly maintains a connectable BLE advertisement and waits for responder-initiated GATT interaction. Android's BLE guidance describes BLE as designed for significantly

lower power consumption, and its advertising API defines low-power advertising as the default and preferred mode because it consumes the least power.[13, 14] Smartphone BLE measurements likewise report that advertiser energy is materially lower than scanner energy; in the tested setup, advertising consumed only a fraction of a mAh over 10 minutes, while the scanner was unlikely to exceed 3 mAh over the same interval.[7] This supports treating SOS BLE standby as the default civilian emergency posture when the app is waiting to be found rather than continuously relaying public traffic.

Chunking and outbound buffering semantics. BLE writes/notifications are constrained in payload size (commonly 20 B at default MTU, and up to 244 B with negotiated MTU). To support larger encrypted payloads, the implementation uses a length-prefixed transport packet (2-byte big-endian length header followed by ciphertext blob) and reassembles chunks before decryption.

On the client side, chunk writes include bounded retries and small pacing delays; on the server side, partial packets are discarded on disconnect.

In SOS chat, outbound payloads are cached per connected peer and delivered through notifications when the peer enables CCCD; characteristic read also exposes the latest payload. This is endpoint-local buffering, not relay/store-and-forward routing.

Message types. The protocol defines message types for text messages, location updates, acknowledgements (ACK), handshake/control messages, and voice-session control.

Example message flow.

1. Discovery: devices detect each other via BLE advertising/scanning.
2. (Optional) authentication: a lightweight BLE handshake verifies context (e.g., Rescue Mode responder verification).
3. Transport setup: a point-to-point RFCOMM link is established.
4. Session setup: a session key is selected/derived (PSK-derived or negotiated after responder verification).
5. Transfer: the sender transmits AEAD-protected frames (ivB64 + ciphertext payloads in baseline text paths).
6. Acknowledgement: the receiver validates and decrypts locally, then returns an ACK for the message ID.

Acknowledgements, timeouts, and retry policy. The sender records a local delivery state (queued, sent, acknowledged).

Implementations define T_{ack} , N_{max} , and a backoff strategy (e.g., exponential with jitter), and report these values alongside pilot results.

Transport and reliability parameters (baseline reference table):

Connection management. The transport is responsible only for byte delivery; the application protocol defines reconnection behavior, idempotence rules, and rate limits to mitigate flooding.

Flow	Core parameters	Failure / recovery behavior
Discovery + contact activation	Device-name discovery: 15 s primary, 8 s fallback; post-pairing handshake timeout: 10 s.	Contact is not activated unless encrypted challenge-response completes within timeout.
Secure text messaging	SEC_MSG + ACK; T_{ack} and N_{max} are build-defined and must be declared in pilot evidence.	Exponential backoff with jitter; duplicate UUIDs dropped; sender state remains queued/sent until ACK or retry budget exhaustion.
Rescue BLE secure chat	MTU-driven chunking (20–244 B notifications), max secure packet 32,767 B, bounded write retries with pacing delays.	Partial packets are discarded on disconnect; peer must reconnect and resume secure exchange after role-gate success.
Voice clip transfer	Chunk 512 B, window 8, ACK interval 32, timeout 3 s, max retry 5.	SHA-256 end verification; verify-fail paths trigger resume/retry sequence.
Image transfer	Dynamic chunk size (512 KB–1 MB), window 12, ACK interval 24, timeout 3 s, max retry 5.	CRC32 per chunk + final SHA-256 verification; corrupted/missing ranges re-requested via resume flow.
File transfer	Dynamic chunk size (128–768 KB), window 10, ACK interval 20, timeout 3 s, max retry 5.	CRC32 per chunk + final SHA-256 verification; resume from last valid acknowledged offset.

7.9 Why point-to-point remains default (with controlled responder mesh extension)

Crisis Connect keeps point-to-point (P2P) behavior as the default communication model for civilian and mixed-use flows on the local radio path, even though v1.1.7 includes both a controlled responder-only Wi-Fi Aware secure mesh extension and an opt-in Public GATT general channel. This preserves operational determinism for the widest user set while allowing specialized coordination modes where governance controls exist. The internet drain (Section 8) does not change this analysis: it is not a radio relay topology, and when infrastructure is down the arguments below apply unchanged.

- Operational simplicity: generalized mesh routing introduces additional roles and state (routing, store-and-forward, duplicate suppression, loop avoidance, rapidly changing

topology). In disaster environments this increases complexity and the number of failure modes. P2P preserves a clearer execution path.

- **Reliability under churn:** In disasters, people and teams move continuously and RF conditions can be unstable (metal/concrete, multipath, congestion). Mesh performance depends on topology quality, which can degrade quickly. P2P provides a clear operational decision: is the peer in range or not?
- **Energy constraints:** Mesh designs often drift toward always-on behavior (relay eligibility, scanning, advertisement, forwarding), which can be costly when charging is scarce. P2P enables an “activate-when-needed” posture and tighter power budgeting.
- **Security and governance:** broad mesh relaying expands the trust boundary (who relays whose data, policy enforcement, metadata exposure). In v1.1.7, responder-secure mesh keeps strict role/certificate gates, while the Public GATT channel remains explicitly opt-in and should be treated as a non-confidential coordination path.
- **Fit to disaster operations:** In earthquake response, early critical communication is often local (nearby survivors, adjacent teams, approaching search-and-rescue units). Under this operational assumption, P2P range can be sufficient for many high-priority interactions.

Future studies may evaluate managed relay/store-and-forward policies for responder fleets. Any such expansion requires explicit routing policy, strict resource limits, and approval by operating authorities.

7.10 Implementation protocol profile (Android baseline v1.1.7)

The Android transport implementation uses line-delimited application frames over RF-COMM. The baseline control and message families are:

- **Handshake families:** `HSK_REQ|challenge|ivB64|cipherB64` (challenge + IV + ciphertext only, no key-carry field) and `HSK_ACK|challenge|ivB64|cipherB64`; timeout 10 seconds.
- **Contact metadata sync:** `CONTACT_INFO|ivB64|cipherB64` (encrypted metadata; key fields are non-authoritative and not used as standalone trust bootstrap).
- **Text message (secure path):** `SEC_MSG|messageUuid|ivB64|cipherB64`.
- **Delivery/read semantics:** `ACK|messageUuid` and BLE ACK payloads (DELIVERED, READ) in Rescue Mode.
- **Backward compatibility:** legacy `MSG|...` and legacy voice prefixes are accepted for interoperability.

JSON frame families are used for richer sessions:

- Voice control frames: `voice:init`, `voice:chunk`, `voice:ack`, `voice:resume`; verification outcomes `voice:verify_ok` and `voice:verify_fail`.
- Image control frames: `image:init`, `image:chunk`, `image:ack`, `image:resume`; verification outcomes `image:verify_ok` and `image:verify_fail`.
- File control frames: `file:init`, `file:chunk`, `file:ack`, `file:resume`; verification outcomes `file:verify_ok` and `file:verify_fail`.
- Call control/audio: `call:offer`, `call:ring`, `call:accept`, `call:reject`, `call:end`, `call:cfg`, `cfg_ack`, `call:audio`, and `audiox`.

1. Discovery and link setup

BLE discovery/control signals peer availability; RFCOMM transport is established for payload exchange.



2. Session activation

Peers execute `HSK_REQ|challenge|ivB64|cipherB64/HSK_ACK|challenge|ivB64|cipherB64`; contact state is activated only after challenge-response validation within timeout.



3. Secure transfer

Sender emits `SEC_MSG|messageUuid|ivB64|cipherB64`; receiver validates AEAD, decrypts locally, and persists by UUID idempotence rules.



4. Delivery progression

Receiver returns `ACK|messageUuid` (and read-state updates when applicable); sender transitions queued → sent → acknowledged.



5. Media reliability branches

Voice/image/file flows invoke `verify/resume` branches on integrity failure or interrupted transfer, bounded by retry policy.

Figure 1: RFCOMM application protocol lifecycle (diagram version DGM-RFCOMM-1.0; legend: top-to-bottom arrows indicate mandatory stage progression in a point-to-point session).

7.11 Media and document transfer profiles

Beyond text and optional voice call sessions, the baseline includes reliable chunked transfer profiles for voice message clips, images, files, and compact offline map-share packages. These transfer profiles are explicitly endpoint-scoped reliability paths; they do not imply uncontrolled relay or store-and-forward networking.

Channel	Implementation profile (current)
Voice message transfer	Default chunk size 512 B, window 8, ACK interval 32 chunks, timeout 3 s, max retry 5, SHA-256 end verification.
File transfer	Dynamic chunk sizing (bounded 128–768 KB), window 10, ACK interval 20, timeout 3 s, max retry 5, CRC32 per chunk and SHA-256 final verification.
Image transfer	Dynamic chunk sizing (bounded 512 KB–1 MB), window 12, ACK interval 24, timeout 3 s, max retry 5, CRC32 per chunk and SHA-256 final verification.
Offline map share	Zipped offline map DB package, bounded to 10 MB for transfer payload construction, then merged on receiver.

7.12 Authorized mesh profile (Wi-Fi Aware secure group chat, v1.1.7)

The responder mesh extension uses a dedicated Wi-Fi Aware service namespace (`com.auralis.crisisconnect.mesh.v1`) and authenticated message families (`hello`, `authOk/authReject`, `chat`).

- Mesh activation policy: disabled by default; optional “mesh always on” is available only to authorized responder roles.
- Eligibility gate: platform support (Wi-Fi Aware capability, required nearby/location permissions, Wi-Fi enabled) and valid local role-certificate posture.
- Message confidentiality/integrity: AES-256-GCM per message, random 96-bit nonce, message-bound AAD (`mesh-chat | messageId | senderNodeId | timestamp`).
- Session key derivation: ephemeral ECDH + HKDF-SHA-256 with peer-node transcript binding (`mesh-chat-v1 | nodeA | nodeB`).
- Scope boundary: secure chat is local-radio responder coordination; it is not a general public mesh relay channel.

Mesh extensions on the v1.1.7 line.. Three additions extend the authority mesh without changing its trust posture:

- **Media accelerators (Wi-Fi Aware / Wi-Fi Direct).** Encrypted media blobs always ride the BLE mesh as the baseline; capable device pairs may *additionally* push the same blob over a Wi-Fi Aware or Wi-Fi Direct fast lane behind a lane-selector abstraction. The Aware datapath PSK is derived from the authority group key, payloads remain group-key AES-GCM end-to-end, and receivers de-duplicate by blob identifier — the accelerator changes throughput, not confidentiality.

- **Mesh push-to-talk (“telsiz”).** Walkie-talkie-style voice over the authority mesh with strict single-speaker floor control and deterministic claim resolution, Opus encoding with jitter-buffer concealment, and multi-hop-aware participant addressing keyed by per-device origin identifiers rather than BLE hop addresses, with join heartbeats for presence.
- **Mesh profiles.** A profile/registry layer scopes mesh services so authority deployments can run distinct mesh configurations without cross-talk, and a history restorer rebuilds mesh chat state across service restarts.

8 Hybrid Internet Transport Plane

The 1.1.x line introduces an internet transport that operates alongside — never instead of — the Bluetooth incident path. This section documents that plane as a source-reviewed implementation baseline (Android v1.1.7, iOS 1.1.2, and the interoperable web messaging surface). It is the largest architectural change since the original whitepaper, and it deliberately preserves the system’s two founding constraints: plaintext never leaves end-user devices, and internet failure must leave the offline path untouched.

8.1 Queue model: internet as a third drain

The chat store-and-forward queue that previously drained over RFCOMM and BLE/GATT gains a third drain: when validated internet connectivity exists and the Bluetooth link to that contact is not currently up, queued messages for internet-capable contacts are sealed into end-to-end encrypted envelopes and relayed through backend infrastructure — Bluetooth remains the preferred drain whenever its link is live. A connectivity-triggered flush sends queued traffic when internet returns. The conversation’s local session code doubles as the cross-device conversation identifier, so a chat started over Bluetooth continues over the internet and vice versa, with one merged timeline. Low-bandwidth operation is a design goal: predefined crisis phrases collapse to one-byte template codes inside the ciphertext, easing base-station load during partial outages, and attachments travel as bounded encrypted chunks.

8.2 Internet identity and registration boundary

- **Identity key.** Each device holds a long-term P-256 identity key: generated inside the Android KeyStore with agreement-only purpose on API 31+ (non-exportable; ECDH executes in-keystore), and in the Secure Enclave on capable iOS hardware. Only the public key is published.
- **Anonymous by default.** On first start the app signs in to the backend anonymously, publishes the identity public key *non-discoverably*, and registers a push token. No account, phone number, or login screen is required; a QR-added contact can reach the

device online without the user ever creating an account, and anonymous devices are reachable only by peers who scanned their QR — never by directory enumeration.

- **Discoverability is opt-in.** Directory lookup by phone number or username requires an explicit, skippable onboarding step with phone verification; the backend indexes a phone number only when it matches the verified authentication claim (directory-poisoning defense). Contact discovery from the address book is matches-only and queries are not stored server-side.
- **Governance note.** This default-on registration boundary (anonymous sign-in, key publication, push token) is a material privacy-relevant behavior and is inventoried in the Privacy Considerations section. QR pairing remains the most private path.

8.3 End-to-end encryption profiles

Two envelope generations are in service, byte-compatible across Android, iOS, and web (enforced by shared golden test vectors):

- **v2 — authenticated ECIES envelope.** Per-message ephemeral P-256 key; the message key is derived via HKDF-SHA-256 over both the ephemeral-static and static-static ECDH shares, giving cryptographic sender authentication (the relay cannot forge messages); AES-256-GCM with sender, recipient, and conversation identifiers bound as associated data (cross-conversation replay defense). Template codes and attachment chunks ride inside the ciphertext.
- **v3 — Signal-protocol sessions.** Current builds prefer Signal sessions via libsignal: PQXDH key agreement (post-quantum, Kyber-based)[15] plus Double Ratchet forward secrecy,[16] prekey publication with automatic top-up, TOFU identity pinning with explicit “security code changed” re-verification flow, and numeric-fingerprint safety numbers surfaced in the chat UI. All send paths funnel through a single seal-and-relay routine that prefers v3 and falls back to v2; once a v3 session exists with a peer, older envelope versions from that peer are refused (anti-downgrade pin). Receive paths fail closed. Sessions address each peer at a single fixed device slot (one logical Signal device per account); multi-device fan-out is not part of the current baseline, so a replacement or additional device presents as a key change subject to the re-verification flow.

8.4 Relay backend and push delivery

The relay is a set of backend functions plus a document store: an identity directory (publish/lookup), a message relay, push-token registration, and delivery fan-out. Its governance-relevant properties:

- **Ciphertext-only.** Relay documents carry routing fields and sealed envelopes; the server can route but never decrypt. A plaintext routing-only ring marker exists solely to trigger call push priority and carries no content.

- **Deletion semantics.** Recipients acknowledge messages to delete relay ciphertext; unacknowledged envelopes expire on a TTL and are purged by scheduled cleanup.
- **Push wake-up.** Delivery fan-out is platform-split. Android tokens receive silent data-only pushes: the client decrypts locally and renders its own notification, because the server holds no plaintext to display. iOS tokens receive a visible alert push whose title/body are generic, device-localized placeholder keys (the server still has no plaintext) carrying the same encrypted data payload, because iOS cannot execute code on a background push once the app is force-quit; a running iOS app processes the payload inline exactly as before. Call ring on iOS additionally uses a direct APNs HTTP/2 VoIP path. The direct APNs VoIP path activates only once APNs credentials are provisioned server-side and is dormant otherwise; deployments expecting iOS call-ring wake-up MUST include that provisioning step in the deployment checklist.
- **Hosting region.** The managed relay deployment currently executes in a single United States cloud region (`us-central1`); ciphertext envelopes, the identity directory, push-token records, and SOS-uplink observations are therefore processed under that region's jurisdiction by default. Deployments with data-residency obligations (e.g., KVKK/GDPR) MUST treat backend region selection — or migration to the institution-controlled hosting option described in the Cloud Services section — as an explicit deployment decision.
- **Abuse resistance.** Backend attestation (Play Integrity / App Check with App Attest on iOS) gates the callables; the server stamps an unforgeable sender identifier on relayed documents; and a claim-before-decrypt rule deduplicates deliveries so duplicates cannot burn one-time prekeys.
- **Graceful degradation.** Backend bring-up is time-bounded, with backoff and retry triggered by connectivity changes; a failed internet bootstrap leaves Bluetooth operation unaffected.

8.5 Authority channel communications

Agency (kurum) users gain channel and hierarchy messaging interoperable with the web control plane:

- Per-agency channels and cross-panel (province-headquarters, sibling-panel) 1:1 hierarchy messaging, rendered natively in the mobile home list and byte-compatible with the dashboard implementation.
- Channel confidentiality uses a shared agency key (AES-256-GCM) issued through a membership-gated backend callable; hierarchy channels use server-issued channel keys with the relationship verified server-side; backend collections hold ciphertext only, and key collections are server-only (not client-readable).
- Encrypted attachments and voice notes: sealed under the channel key with associated data bound to the channel; blobs are stored encrypted in object storage with encrypted

descriptors in the message document; offline-first local persistence with an on-device media cache.

- Channel calls reuse the WebRTC engine, signaled over a membership-gated collection, interoperable with the web dashboard.
- **Offline Bluetooth bridge.** Channel peers are auto-linked as hidden bridge contacts over the citizen dual-mode pipeline (number-keyed SPAKE2 plus RFCOMM). When the internet cannot carry the channel, text, locations, media, and voice calls ride Bluetooth with a visible “Connected via Bluetooth” badge, and Bluetooth-era messages backfill to the channel when connectivity returns. Authority messaging therefore survives internet loss — the offline-first inversion is deliberate and user-visible.

8.6 SOS uplink and contact escalation

The BLE SOS broadcast remains the primary, offline-first distress channel and never waits on cloud code. When connectivity exists, two additive behaviors apply:

- **Cloud uplink.** The active SOS is self-reported to the authority dashboard through a backend callable and refreshed on a battery-aware heartbeat (approximately 45 s normally, 90 s on low battery), with store-and-forward retry for offline stop/resolve transitions.
- **Contact escalation.** On SOS start, the top internet-capable contacts automatically receive an SOS alert template with a map link (user-designated emergency contacts take precedence; otherwise an automatic most-messaged/longest-known ranking applies; current Android builds notify up to five contacts and current iOS builds the top three); movement beyond approximately 25 m streams live-location updates that rewrite the same message bubble; stopping SOS sends an all-clear, persisted and retried if offline.

Governance treatment is unchanged: dashboard-side SOS records are time-windowed operational observations, not verified casualty truth, and the uplink is operational metadata governed separately from message content.

8.7 Internet calls (WebRTC) and cross-platform Bluetooth calls

- **1:1 internet calls.** Voice and video calls (plus screen share into a call) run over WebRTC between contacts.[17] Offer/answer and ICE candidates ride the same sealed E2E relay envelopes (short-TTL call templates); the offer itself is the ring with bounded ring time, busy handling, and bounded ICE-restart recovery. Media is peer-to-peer where possible, with per-session TURN relay credentials minted server-side (Cloudflare Realtime anycast TURN; anonymous/QR-only accounts mint relay credentials too, and a STUN-only fallback — currently Google public STUN servers, a provider-dependency inventory item — keeps direct peer-to-peer calls possible if the credential fetch fails). Platform

integration uses the Android Telecom framework and iOS CallKit, with PushKit VoIP wake-up for force-quit iOS apps.

- **Bluetooth GATT calls.** Fully offline voice calls now also run over the encrypted point-to-point GATT link, enabling Android-iPhone calls without any infrastructure. Signaling rides the per-contact AES-GCM chat envelope as dedicated call kinds (silently ignored by older versions); audio is Opus in a compact binary frame encrypted with AES-256-GCM under HKDF-derived directional keys with fresh per-direction salts (nonce-reuse defense), sized to single ATT writes at MTU 247, with receiver-driven bitrate adaptation from 12 kbps. Android-Android calls remain RFCOMM-first with automatic GATT fallback.
- **SFU group calls with MLS.** Multi-party calls use a selective forwarding unit (Cloudflare Realtime) reached through a server-side proxy that keeps the SFU secret out of clients. Media frames are end-to-end encrypted per-frame with an MLS group key[18] (OpenMLS-based shared Rust core compiled for web, Android, and iOS); the MLS handshake relays as opaque blobs, group-creator claim is transactional (hijack defense), room listing is deny-by-default with room ID as capability, and safety numbers surface in the call UI. **Status discipline:** the capability is live on the web line and Android development line, and fully built but feature-gated *off* on iOS pending two-device and three-way interop validation. Reviewer materials must not claim it as a shipped iOS feature until the gate flips. One dependency-fragility note: on iOS the per-frame encryptor attaches through non-public (ABI-verified, not publicly stable) frame-encryptor entry points of the prebuilt WebRTC binary, so any WebRTC dependency upgrade **MUST** re-run the automated ABI gate before the feature flag is enabled.

8.8 Internet-plane evidence posture

Consistent with this document's evidence discipline, the internet plane is currently supported by source review, cross-platform golden-vector test suites, and developer-recorded interop checks (for example, two-way Android-iOS GATT call audio against the Android 1.1.7 store build). It does not yet have a reviewer-grade field packet. Pilot planning **MUST** therefore define internet-plane gates analogous to the offline gates: relay delivery success and latency under degraded networks, push wake-up reliability across OEM power policies and iOS background states, TURN fallback behavior, battery impact of hybrid standby, and safety-number verification workflow evidence. These appear as new claims and open gaps in the Evaluation Plan.

9 Cryptographic Design

9.1 Threat model summary

We assume an attacker can passively listen to local wireless traffic and may attempt active interference (spoofing, replay, denial-of-service). On the internet path, we additionally treat the relay backend, push infrastructure, and TURN relays as honest-but-curious at best and potentially compromised at worst: confidentiality and sender authenticity **MUST NOT** depend on server behavior, which is why the envelope profiles authenticate senders cryptographically and current builds add forward secrecy. We do not assume the attacker can compromise the user device OS or extract secrets from hardware-backed storage. A full threat model is provided in the Threat Model section.

9.2 Contact communication (pre-shared keys)

For contacts provisioned via QR exchange, communication uses a pre-shared symmetric key and authenticated encryption (AES-GCM profile).[12] In baseline operation, trust setup remains QR-first. In the hardened v1.1.7 profile, `HSK_REQ|challenge|ivB64|cipherB64` no longer carries key material. Key recovery remains out-of-band or auditable administrator-driven.

9.3 Responder authentication (certificates)

For Rescue Mode interactions, responder identity is authenticated using public-key certificates (e.g., ECDSA).[19] Victim devices verify responder certificates offline using a pinned authority public key embedded in the app.

Implementation-conformance note (v1.1.7). The current Android + cloud baseline issues role certificates with bounded validity (current issuer profile: 72-hour TTL), stores them locally, and validates signature, owner UID, role scope, and time window offline against the pinned master public key. Certificate cache usage is additionally bound to the current device public key and active user UID; if no usable cached certificate exists, responder-secure flows require at least one online verification cycle.

Revocation scope clarification. CRL/denylist snapshot ingestion is documented as an operational governance target, but an explicit on-device CRL parser/enforcement path is not part of the validated v1.1.7 Android baseline.

iOS conformance note (1.1.2). On iOS, role-certificate provisioning binds Apple App Attest attestation to a Secure Enclave device key, with the same 72-hour validity profile and automatic background renewal (a scheduled background task renews certificates with less than 24 hours remaining). On Android, non-Firebase backend HTTPS is routed through a pinning-capable shared HTTP client backed by a Network Security Config that disables cleartext globally and excludes user-installed CAs in release builds; SPKI pin sets are scaffolded as mandatory policy for custom hosts but are not yet populated for any endpoint.

9.4 Responder mesh session security (v1.1.7 extension)

For authorized responder mesh sessions, peers exchange signed role proofs and ephemeral session public keys, then derive per-peer symmetric keys using ECDH P-256 and HKDF-SHA-256. Chat payloads use AES-256-GCM with message-bound associated data for confidentiality and integrity.

9.5 Internet-path cryptography (hybrid baseline)

The internet plane's mechanisms are specified in Section 8; the profile summary for review purposes is:

- **Envelope v2 (ECIES):** P-256 ECDH[20] combining ephemeral-static and static-static shares → HKDF-SHA-256 → AES-256-GCM; sender/recipient/conversation identifiers bound as AAD; per-message ephemeral keys; sealed-sender authenticity without trusting the relay.
- **Sessions v3 (Signal):** PQXDH post-quantum key agreement (Kyber/ML-KEM family[21]) plus Double Ratchet via libsignal; prekey pool with signed and last-resort Kyber prekeys; TOFU pinning, anti-downgrade enforcement, fail-closed receive, numeric-fingerprint safety numbers.
- **Identity custody:** P-256 identity keys are hardware-backed where the platform allows (Android KeyStore agreement-only keys on API 31+; iOS Secure Enclave) with software fallback on older hardware.
- **SPAKE2 bootstrap:** RFC 9382 P-256/SHA-256/HKDF/HMAC profile, pinned to the RFC test vectors for cross-platform byte-compatibility; identity exchange under the derived key is AES-256-GCM. Disclosure: the SPAKE2 core, including its P-256 group arithmetic, is an in-house implementation rather than a vetted third-party library; correctness is vector-pinned, but independent implementation review (including side-channel and constant-time posture) SHOULD be an explicit item in the pre-pilot cryptographic assessment.
- **Group media (MLS):** per-frame media encryption under an OpenMLS-derived group key for SFU calls; the forwarding unit and coordination store carry only ciphertext and opaque handshake blobs.
- **Call audio (GATT):** HKDF-derived directional keys from the contact key with fresh per-direction salts per call instance; AES-256-GCM per audio frame.

Automated coverage for these paths includes cross-platform golden vectors (envelope v2, Signal v3 codec, call packets, MLS framing, agency channel crypto) maintained in the client repositories; reviewer packets SHOULD attach the corresponding test-run exports.

9.6 Cryptographic profile lock (pilot gate)

To reduce ambiguity during institutional review, pilot acceptance is tied to a fixed cryptographic profile unless an approved exception is documented in the evidence packet.

Control element	Locked pilot profile
Contact-message AEAD	AES-256-GCM with 96-bit IV and 128-bit authentication tag.[12]
Responder mesh chat AEAD	AES-256-GCM with 96-bit IV; AAD binds message ID, sender node ID, and timestamp for authenticated decryption context.
Responder key agreement	ECDH over NIST P-256; session keys derived using HKDF-SHA-256 with transcript binding.[19, 22]
Responder signatures	ECDSA P-256 with SHA-256 for role-proof verification.[19]
Internet envelope (v2)	Authenticated ECIES: P-256 ECDH (ephemeral-static + static-static) with HKDF-SHA-256 and AES-256-GCM; sender/recipient/conversation AAD binding; per-message ephemeral keys.
Internet sessions (v3)	Signal protocol via libsignal: PQXDH (Kyber) + Double Ratchet; TOFU pinning with anti-downgrade enforcement; numeric-fingerprint safety numbers; fail-closed receive.
Group-call media (MLS)	Per-frame AEAD under an OpenMLS group key for SFU calls; SFU and coordination store hold ciphertext and opaque handshake blobs only.
Nonce uniqueness governance	CSPRNG-generated 96-bit IV per encrypted packet, bounded packets-per-key policy, and mandatory key refresh on restart/re-handshake; GATT call audio adds per-direction HKDF salts per call instance.
Replay/duplicate defense	Receiver replay cache over sender/message UUID tuples + persistence-layer UUID uniqueness constraints.
Production key recovery rule	No plaintext/unauthenticated in-band key-carry; recovery is limited to QR out-of-band reprovisioning or auditable administrator provisioning.

Implementation key custody and lifecycle (Android baseline). Responder-identity signing keys are generated under `AndroidKeyStore` alias `dcs_device_identity`; the private key is non-exportable and used in-keystore for signature operations. Local symmetric material used for app-level key state and SQLCipher database passphrase seed is stored in `EncryptedSharedPreferences` (AndroidX Security, `AES256_SIV/AES256_GCM`), and the encrypted Room/SQLCipher database key is derived from that protected store

at runtime. Pilot lifecycle policy is key-refresh on restart/re-handshake for secure sessions, certificate cache binding to current public key + owner UID, and mandatory reprovisioning when key/certificate mismatch or expiry is detected; Android manifest backup remains disabled (`allowBackup=false`).

10 Rescue Mode

Rescue Mode supports two distinct functions: (i) broadcasting a local distress beacon and (ii) enabling authenticated two-way communication with verified responders.

10.1 SOS beacons

A device in SOS state emits connectable BLE advertisements for the Rescue service UUID. In the current Android baseline, advertisement content is minimal (service UUID signaling); authenticated identity exchange and secure payload handling occur after GATT connection and handshake.

When internet connectivity exists, the active SOS is additionally self-reported to the authority dashboard on a battery-aware heartbeat and escalated to top internet-capable contacts, as documented in Section 8; the BLE beacon remains the primary channel and never waits on cloud code. Recent baselines also add a user-facing arming countdown before broadcast, a policy-flagged child profile mode, and a region-aware emergency-call button that resolves the locally applicable official emergency number (for example 112/911) — reinforcing the design position that official channels are attempted first when any telephony survives.

10.2 Responder interaction

When a responder connects, the victim device performs an authentication handshake over BLE GATT. Only after successful verification does the system establish a session key for subsequent secure messaging.

10.3 BLE GATT profile (implementation baseline)

The Rescue Mode GATT profile is implemented as a dedicated primary service with assigned-number-derived UUIDs:

Operational constraints in the current build:

- Max role-proof transport packet: 512 B.
- Max secure chat packet (after reassembly): 32,767 B.

Identifier	Purpose
Service 0xCC00	Rescue Mode primary service (connectable advertisement target).
0xCC10/0xCC11	Auth challenge / auth response for ECDH-based session key setup.
0xCC20/0xCC21	Secure inbound role-proof channel and ACK channel.
0xCC30/0xCC31	Secure chat in / secure chat out characteristics.

- Default MTU assumption: 23; notification chunking scales with MTU and is bounded to 20–244 B per notification.
- ACK payload semantics: OK (handshake), DELIVERED, READ.

Role-proof payload and validation rules (baseline policy).

- Role-proof payload MUST include responder certificate reference, role scope, issuance time, and explicit expiry.
- Payload MUST be transcript-bound (challenge nonce and peer ephemeral key context) before signature verification.
- Verification pipeline MUST check signature validity, certificate chain/trust anchor, expiry at handshake time, and role authorization for Rescue chat.
- Expired, malformed, or scope-mismatched proofs MUST be rejected before secure-chat promotion.

Only verified responders are promoted to the secure-chat path; unverified clients are rejected at the secure write gate.

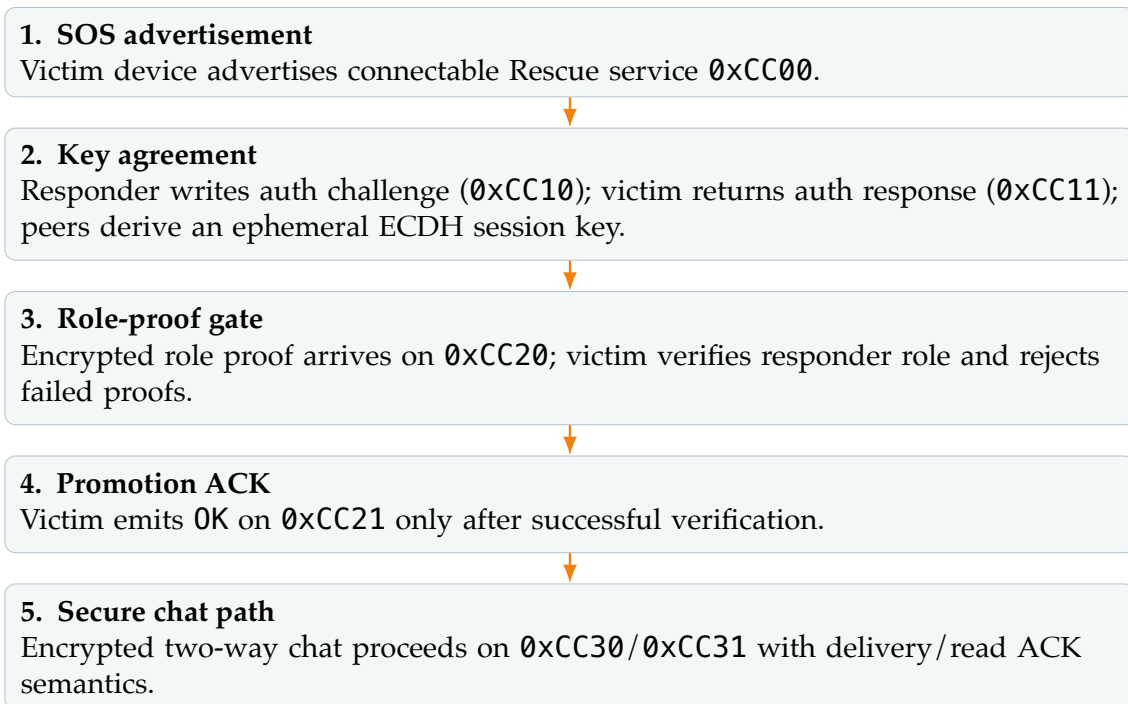


Figure 2: Rescue Mode BLE GATT authentication and secure-chat transition (diagram version DGM-GATT-1.0; legend: top-to-bottom arrows indicate hard promotion gates that must pass before secure chat).

11 Voice Communication

Voice is implemented on three paths: (i) real-time peer-to-peer streaming over the point-to-point RFCOMM transport (Android–Android, documented in this section), (ii) encrypted Bluetooth GATT calls that also work cross-platform (Android–iOS), and (iii) internet WebRTC calls with optional video and screen share — the latter two are specified in Section 8. On every path, audio is encoded with Opus and encrypted prior to transmission.[23] Voice support is optional and should be evaluated carefully due to power and bandwidth constraints.

11.1 Bandwidth and quality considerations (implementation configuration)

Opus supports a wide range of bitrates and modes; this deployment therefore documents the concrete voice configuration used in evaluation.[23]

Opus configuration (current).

- Sampling rate: 16 kHz.
- Frame size: 10 ms.

- Packetization: 5 frames per packet (approximately 50 ms packetization interval).
- Bitrate targets: 20 kbps (wideband) and 16 kbps (narrowband).

Opus bitrate vs. RFCOMM capacity. RFCOMM must provide an effective throughput comfortably above the selected Opus bitrate plus packetization/transport overhead (headers, framing, retransmissions, and encryption/authentication overhead). If the available RFCOMM capacity drops near or below the codec rate, the sender queue grows and jitter/loss increase, which directly degrades both voice quality and latency.

Latency targets (VoIP). With approximately 50 ms packetization and a jitter buffer configured to 20 ms to 50 ms (default 30 ms), the design target for buffering-driven one-way delay is roughly 70 ms to 100 ms. In end-to-end measurements, additional device audio pipeline and scheduling delays must be accounted for and reported.

11.2 Measurement requirements

To avoid over-claiming, voice should be reported with device models and explicit test conditions. At minimum, pilots should measure call setup time, sustained bitrate, packet loss (if observable at the application layer), end-to-end latency distribution, and incremental battery drain relative to idle discovery.

11.3 Concurrency

The baseline design assumes a single active voice session per point-to-point link. Supporting multiple concurrent voice sessions on a single device is out of scope for the baseline and would require explicit resource management and scheduling under constrained power.

11.4 Adaptive voice profile switching (implementation policy)

To improve call continuity under degrading radio conditions, the implementation applies bounded adaptive switching between wideband (WB) and narrowband (NB) operating profiles.

- Control loop period: 1 second.
- Cooldown between profile changes: 6 seconds.
- WB-to-NB trigger (degrade): moving average write time above 25 ms, or repeated play-out underruns, or persistently shallow jitter queue.
- NB-to-WB trigger (recover): moving average write time below 10 ms, zero underruns over sustained windows, and stable queue depth.

- Safety behavior: all profile changes are negotiated with call config ACK frames before activation.

This policy should be treated as an implementation baseline, not as a universal QoS guarantee.

12 Background Services

The system relies on OS-conformant background / foreground services to maintain readiness. This includes a transport listener (RFCOMM), BLE discovery / SOS services, optional responder mesh coordination, and optional Crisis Link telemetry sync.

12.1 Android constraints

Android background execution limits and OEM power management can affect scanning frequency, advertisement continuity, socket liveness, and Wi-Fi Aware session continuity. Institutional deployments should therefore document: (i) minimum supported Android versions, (ii) required foreground-service modes, (iii) Wi-Fi Aware capability coverage across device fleets, and (iv) any device-model-specific restrictions observed during trials.

Android's BLE background guidance also notes that background discovery and connection flows still depend on the app process being alive, and long-lived communication tasks generally require a foreground service.[24] This interacts directly with power policy: victim-side SOS advertising / server standby can remain comparatively light, but responder scanning, public relay, and Wi-Fi Aware sessions keep more radio state active and are therefore the parts that should be tightly policy-gated.[6, 4]

12.2 iOS development status and readiness gates

The iOS repository is a standalone client application rather than a thin companion. The current Xcode project declares app version **1.1.2** for bundle identifier **com.auralis.crisisconnect**, targets iOS 17 on both iPhone and iPad, uses SwiftUI for the primary product surface, includes a full Android-parity onboarding flow with optional phone verification, and organizes the user experience into Messages, Tools, Survival Guide, and Settings areas. Since June 2026 the line has been developed as an explicit *parity program* against the shipping Android app: roughly forty commits are tagged as Android-parity work, and wire/crypto byte-compatibility with Android and web is enforced through shared golden test vectors (internet envelope, Signal session codec, call packets, MLS framing, agency channel crypto). Release engineering moved from a manual archive workflow to a Codemagic CI pipeline with App Store Connect integration that builds signed IPAs and publishes to TestFlight; the app-target build number reflects real CI history. For document traceability, this edition refers to the iOS **1.1.2** line as the current *store-listed parity implementation baseline*. That phrase is

justified by the repository-visible version, CI configuration, feature surface, and parity test suites; it is not equivalent to the validated institutional deployment baseline, which still remains Android v1.1.7 until iOS field gates are closed. One release-control caveat is reviewer-relevant: the CI release workflow triggers on the main branch, while the newest feature work (internet calls, Signal v3, PushKit wake-up, expanded localization, authority messaging) lives on a feature branch that has not yet been merged to the release branch — so the newest capabilities have not yet passed through the TestFlight pipeline. As with the web line, the iOS repository carries no git tags; review claims **MUST** bind to the exact source commit and branch rather than to the marketing version alone.

iOS baseline dimension	Repository-visible position
Documented iOS baseline	Standalone iOS 1.1.2 store-listed parity implementation baseline in <code>Crisis Connect.xcodeproj</code> ; institutional deployment baseline remains Android v1.1.7.
Package identity	Bundle identifier <code>com.auralis.crisisconnect</code> .
OS and device coverage	iOS 17 target on iPhone and iPad (<code>TARGETED_DEVICE_FAMILY=1,2</code>); localized in 19 languages (RTL layout for ar/fa/ur follows the device system language automatically; the in-app language override switches strings without re-mirroring the layout).
Primary UI architecture	SwiftUI application with Messages, Tools, Survival Guide, and Settings tabs plus Android-parity onboarding with optional phone verification.
Runtime declarations	Bluetooth central/peripheral, microphone, camera, contacts, local network, location, audio, fetch, processing, VoIP, and remote-notification declarations with background-task identifiers for offline-map continuation, Crisis Link refresh, and certificate renewal.
Release engineering	Codemagic CI workflow with App Store Connect API-key signing and TestFlight publication; manual archive/evidence kit retained as the fallback path; newest feature branch not yet merged to the release branch.
Responder access gate	Rescue tools are role-gated to <code>admin</code> and <code>fieldteam</code> state; role certificates provision via App Attest bound to a Secure Enclave device key with background renewal.

- **Trusted-contact bootstrap.** The QR and BLE pairing flow is concretely implemented in `NewContactView` together with dedicated contact broadcaster and verifier managers: the app can generate QR payloads containing session code, AES key, platform tag, and share identifier; expose a BLE GATT pairing service; validate the scanned peer through HMAC-based hello/finish exchanges; and then persist the contact locally before opening a verified chat session. On the current line, pairing also provisions the peer's internet messaging identity bidirectionally, can fall back to an internet-only contact when Bluetooth verification fails in the field, and SPAKE2 add-by-number bootstrap is implemented with RFC-vector-pinned tests.

- **General communication path.** The offline iOS transport path remains BLE-centric: `P2pGattChatManager` handles encrypted point-to-point BLE GATT chat, read receipts, and chunked voice transfer. The current line adds the full internet messaging plane in parity with Android: anonymous internet identity with Secure Enclave key custody, v2 ECIES envelopes and v3 Signal sessions (vendored `LibSignalClient` v0.86.5 with PQXDH and Double Ratchet; WebRTC ships as a prebuilt Google binary, build 149), template messages, encrypted attachments, receipts/typing/presence, and silent-push delivery with local decryption. At the UI layer, the chat surface includes voice recording, attachment persistence, waveform extraction, playback controls, safety numbers, and key-change warnings.
- **Live voice and video paths.** The current line carries three call workflows: encrypted Bluetooth GATT calls interoperable with Android (Opus in AES-GCM frames with per-direction HKDF keys; `CallKit`-presented), 1:1 internet WebRTC calls with video and inbound screen share (signaling over the sealed relay, TURN fallback, `CallKit` plus `PushKit` VoIP wake-up from force-quit), and the legacy adjacent-device `MultipeerConnectivity` workflow (`ccvoicechat`). SFU group calls with MLS media encryption are fully built but feature-gated off pending interop validation; iOS-originated screen share awaits a broadcast extension.
- **Authorized Rescue / NaN path.** Rescue discovery is a separate, role-gated surface rather than a general-user channel. The Rescue client UI is shown only to authorized responder roles.
- **Authorized transport behavior.** Within that Rescue scope, the manager starts a Wi-Fi Aware / NaN browser and publisher for paired devices first, can retry with user-selected devices through the Wi-Fi Aware pairing card, and keeps BLE Rescue scanning enabled as fallback or parallel discovery. The whitepaper should therefore frame NaN as an authorized responder workflow, not as a universal iOS transport.
- **Institutional access controls.** `FirebaseBootstrapper` performs anonymous bootstrap, writes or refreshes a user document, restores cached AES material, and resolves role state. The security repository and role-proof components show a certificate-backed responder design using Firebase Functions, cached role certificates, signed role proofs, nonce binding, and clock-skew checks. The profile/auth surface also supports Google Sign-In and email/password sign-in, synchronizes country and agency metadata, and exposes role, verification, device-key, and certificate-readiness state in the UI.
- **Platform and security posture.** The codebase provisions Firebase App Check with App Attest when available, falling back to DeviceCheck or debug providers as needed. Device identity uses P-256 private keys with Secure Enclave preference and `AfterFirstUnlockThisDeviceOnly` protection. Keychain storage is used for AES keys, role state, UIDs, broadcast identifiers, device identifiers, and pairing/session codes. Platform runtime checks explicitly disable BLE peripheral hosting in unsupported iPad-on-Mac environments.

- **Offline operational tooling.** The non-messaging surface is also substantive: offline maps include tile download, caching, pause/resume, and local region persistence; the LiDAR tool is positioned as a night-vision-style forward-clearance aid using ARKit scene depth, route cues, obstacle alerts, freeze/live controls, and haptic danger feedback; and the app also exposes sensors, compass, signal scanning, whistle, metal-detection, notifications, language/theme, and privacy settings. This is a broader emergency utility client, not only a messaging shell.
- **Platform-gated NaN implementation.** The repository's Wi-Fi Aware / NaN code path is real, but it is guarded behind `@available(iOS 26.0, *)`. That means the responder-only NaN workflow should be described separately from the general iOS 17 BLE path and should not be collapsed into a single "Wi-Fi + BLE for everyone" claim.

iOS Night Vision / LiDAR obstacle-guidance pipeline.. The Night Vision surface is not merely a cosmetic green camera filter. On supported hardware, `LiDARViewModel` starts an ARKit world-tracking session, requests `smoothedSceneDepth` when available (falling back to `sceneDepth`), and enables mesh or mesh-with-classification reconstruction when supported. The processing loop runs at approximately 5 Hz, evaluates a configurable forward region (*precision, balanced, wide*), and samples left/center/right lanes independently before computing a conservative forward clearance estimate. That estimate is then turned into guidance states (*move forward, move left, move right, stop, scan slowly*), signal-quality grading, and adaptive display depth scaling. The visual output composites a monochrome night-style camera image with an edge-emphasized depth overlay, while the operator HUD exposes distance, guidance, signal quality, and alert radius. Alert presets are source-visible and concrete: the current implementation offers *near, balanced, and early* warning profiles with caution thresholds around 1.0/1.4/1.9 m and danger thresholds around 0.55/0.85/1.15 m. Freeze/live controls and persisted haptic preferences are implemented, with periodic tactile warnings on danger, caution, or signal-loss states. This is a meaningful field-expedient obstacle-guidance aid for darkness, dust, or degraded visibility, but it still must be described as a situational-awareness tool rather than a certified life-safety ranging instrument.

iOS field-utility stack beyond messaging.. The broader iOS tool surface is substantively operational. Offline maps support bounded-region selection, zoom-range control, tile-count and storage estimation, pause/resume, persisted region metadata, and resumable background continuation via `BGProcessingTask`. The download pipeline includes retry/backoff handling, background resumption, and persistent tile/region stores rather than assuming a single foreground session. The sensor dashboard exposes accelerometer, gyroscope, magnetometer, device-motion, altimeter, heading, and proximity signals; the compass and signal-finder surfaces provide fast orientation and nearby BLE radio scanning; and the whistle/metal-detection utilities round out the field-tool set. In reviewer terms, the iOS app is not just a communication shell: it is a multi-tool emergency field client whose platform-specific sensors are deliberately integrated into the product surface.

iOS live voice, rescue runtime, and trust posture.. The live-call implementation on iOS also merits explicit treatment. Nearby iOS voice uses MultipeerConnectivity discovery/advertising, encrypted peer sessions, CallKit presentation, AVAudioEngine capture/playback, and a 24 kHz mono profile; it should be described as a local iOS-to-iOS voice workflow, not as a generic radio primitive available across all transports. Rescue live-location behavior is likewise non-trivial: `RescueLiveLocationCoordinator` selects different runtime profiles for foreground and background operation, can request Always authorization where policy allows, may use significant-location-change monitoring when background authority exists, and queues pending dashboard sync operations for later drain if immediate cloud sync is unavailable. Role access is explicitly constrained to `admin` and `fieldteam`; App Check prefers App Attest, and device keys prefer Secure Enclave-backed P-256 storage with `AfterFirstUnlockThisDeviceOnly` protection. This gives the iOS line a materially professional trust and operations posture, even though pilot evidence is still thinner than the Android packet.

The April 5, 2026 evidence anchor remains in the packet: a passing 43-test plan, a signed Xcode Organizer archive with Apple-upload metadata, first physical iPad deployment and manual workflow evidence, and first basic iPad-to-Android messaging evidence. Since then, the line has moved substantially. The version line advanced to **1.1.2** in alignment with the Android release track, and more than a hundred commits landed between late June and mid-July 2026, dominated by the parity program described above. Materially new since the last edition: the complete internet messaging plane (anonymous identity, v2/v3 envelopes with vendored libsignal, push delivery), 1:1 internet WebRTC calls with CallKit/PushKit integration, cross-platform Bluetooth GATT voice calls, authority channel messaging with the offline Bluetooth bridge, mesh push-to-talk (“telsiz”) over the authority GATT mesh, SOS contact escalation and cloud uplink, Crisis Sentinel with an on-device LiteRT-LM runtime plus an online engine, the built-but-gated SFU/MLS group-call stack, SPAKE2 pairing, 19-language localization, and 17 new unit-test files (19 total in the unit-test target, up from 2 at the April baseline) covering Signal sessions, crypto golden vectors, SFU frame-crypto ABI, SPAKE2, and call protocol behavior. Developer-recorded interop checks exist against the Android 1.1.7 store build for GATT voice calls, internet messaging, and QR pairing. For whitepaper wording, this is strong enough to call the line a store-listed parity implementation baseline under active CI/TestFlight governance — materially beyond the April “signed pilot candidate” label. It is still not strong enough to call the same line the validated institutional deployment baseline: the evidence packet still lacks Android-depth device-matrix and field-run artifacts, the newest feature branch has not passed through the release pipeline, documented parity gaps remain (iOS-originated screen share, the SFU feature gate), and reviewer-grade closure is still missing for BLE continuity, local-voice resilience, cross-platform handshake depth, rescue certificate fault handling, and background-state edge cases on representative hardware.

Remaining readiness gates for iOS should therefore include:

- merging the current feature line to the release branch and shipping it through the CI/TestFlight pipeline so the assessed build and the distributed build are the same artifact;
- repeated iPhone and iPad runs for QR bootstrap, BLE discovery, advertising, reconnection, and lock-state continuity under iOS background policies;
- end-to-end call validation on all three paths — GATT cross-platform calls, internet WebRTC calls (including PushKit wake-up from force-quit, CallKit answer/end flows, route changes, and TURN fallback), and the legacy adjacent-device workflow;
- end-to-end Android interoperability depth for contact pairing, point-to-point chat, internet messaging (v2/v3 including downgrade-pin behavior), voice chunking, SOS authentication, delivery receipts, and role-proof validation flows;
- internet-plane behavior evidence: silent-push delivery reliability across background states, prekey top-up and identity-recovery flows, safety-number and key-change UX validation, and relay acknowledgement/TTL deletion verification;
- explicit product policy for the authorized responder-only Wi-Fi Aware / NaN path, covering paired-device default behavior, manual pairing workflow, role-gated access, and BLE fallback expectations;
- certificate issuance via App Attest, cache reuse, expiry and background-renewal handling, anonymous-bootstrap recovery, and offline-grace validation on representative responder and non-responder accounts;
- an explicit gate-flip protocol for SFU/MLS group calls (two-device plus web three-way interop validation) before the feature is enabled or claimed;
- hardware-coverage documentation separating LiDAR-capable iPhone/iPad models, non-LiDAR devices, and unsupported iPad-on-Mac execution so the Night Vision claim stays operationally precise;
- broader automated and manual test evidence with reproducible pilot artifacts traceable to the same decision gates used for the Android baseline.

Production scope should be expanded only after these milestones are documented with test evidence.

12.3 iOS deployment-baseline promotion standard

For institutional use, the phrase “iOS deployment baseline” should be reserved for a narrower condition than “the app builds and the features exist.” Promotion from the current documented iOS implementation baseline to an institutional deployment baseline should occur only when the institution can point to a fixed release candidate, a reviewer-visible evidence packet, and repeatable pass results on representative Apple hardware. In other words, promotion should be treated as a governance and assurance event, not as a wording preference.

Promotion gate	Minimum standard	Reviewer-visible evidence
Release control	Freeze a single iOS 1.1.2 release candidate with pinned Xcode/toolchain context, dependency snapshot, scheme, bundle identifier, entitlements, and signing posture.	Signed Release IPA or archive metadata, build manifest, checksum/hash manifest, dependency snapshot, and exact source provenance commit.
Automated assurance	Replace template-level test posture with non-trivial unit and UI coverage over the shipping iOS surface. Hard gates should at minimum cover pairing/bootstrap logic, role/certificate handling, Rescue runtime policy, offline-map continuation, LiDAR processing, and primary UI flows.	Exported test plan, pass/fail report, failing-test count of zero on the release candidate, and retained logs/artifacts for the same source line.
Device matrix	Validate on representative Apple hardware classes rather than a single handset. At minimum separate LiDAR-capable iPhone, non-LiDAR iPhone, and iPad behavior; record OS version, battery state, permissions state, and runtime mode.	Device matrix sheet, run ledger, and scenario-to-device traceability table.
Messaging and transport continuity	Demonstrate repeated pass results for QR bootstrap, BLE discovery, verified contact activation, reconnection, lock-state continuity, delivery receipts, and Android interoperability on the release candidate.	Scenario report with pass rate, median/P95 timings where relevant, environment notes, and raw logs or screenshots sufficient for audit replay.
Local live voice	Validate the iOS-to-iOS live-call workflow as its own transport feature: discovery, local-network prompt handling, CallKit answer/end lifecycle, route changes, interruption recovery, and dropout behavior across foreground/background transitions.	Call-flow scripts, route-change logs, setup-time/continuity measurements, and annotated traces for both success and fault-injection runs.

Promotion gate	Minimum standard	Reviewer-visible evidence
Rescue trust and runtime	Close responder-only gates: role issuance, certificate cache reuse/expiry behavior, offline grace, live-location policy, foreground/background location behavior, and NaN policy wording for supported platforms. Any trust failure should block promotion.	Responder account matrix, certificate-lifecycle evidence, live-location traces, signed role-proof validation logs, and explicit NaN/BLE fallback policy note.
LiDAR / field-utility claims	Support Night Vision and tool-surface claims with hardware-aware evidence. LiDAR guidance must be tied to supported-device classes, dark-scene runs, fallback behavior on unsupported devices, and documented alert-threshold policy.	Supported-device matrix, dark-scene or degraded-visibility test notes, unsupported-device screenshots, LiDAR threshold sheet, and retained raw capture examples where allowed.

The promotion decision should also distinguish **hard gates** from **soft gates**. Hard gates are release integrity, trust correctness, responder access control, and transport continuity; failure in any of these should block iOS deployment-baseline designation. Soft gates improve board confidence but may be closed in phased rollout, such as broader hardware breadth, larger sample counts, and richer field artefact packaging, provided the institution explicitly records that residual risk.

iOS deployment-baseline packet.. If the owner wants to claim that iOS has reached deployment-baseline status, the reviewer packet should include all of the following in one traceable bundle:

1. Release manifest: app version, bundle identifier, source commit, dependency snapshot, signing/export method, checksum, and build timestamp.
2. Assurance report: automated-test summary, scenario pass/fail matrix, open defects, and disposition of any waived warnings.
3. Device-and-environment matrix: hardware model, iOS version, LiDAR availability, permission state, battery state, and test setting for each run.
4. Claim-to-evidence map: each iOS-facing whitepaper claim mapped to one or more concrete artefacts, with unsupported claims explicitly removed or narrowed.
5. Known-limitations sheet: unsupported devices, degraded behaviors, remaining warnings, and operational notes for responders, pilots, and institutional reviewers.

Professionally stated, iOS should be promoted to deployment-baseline status only when a third-party reviewer can answer three questions from artefacts alone: “What exact build was assessed?”, “On which Apple devices and conditions did it pass?”, and “Which claims remain intentionally out of scope?” If those answers are incomplete, the correct label remains implementation baseline or pilot baseline rather than deployment baseline.

12.4 Android service lifecycle behavior (validated baseline)

The Android baseline uses dedicated foreground services for long-running communication tasks:

- **RfcommForegroundService.** Classic transport listener, call signaling, media transfer, and the SPAKE2 add-by-number nearby advertising / seeking runtime (number-free beacon; the former standalone **NearbyDiscoveryService** was retired into this service).
- **P2pGattServerService.** Point-to-point GATT chat/call endpoint (dual-mode contact fallback and cross-platform Bluetooth calls).
- **GattMeshForegroundService.** Public GATT mesh coordination runtime.
- **GattSOSServerService.** SOS broadcaster and secure responder interaction endpoint.
- **GattRescueClientService.** Rescue-client BLE session manager.
- **MeshAwareService.**
Authorized responder Wi-Fi Aware mesh discovery / authentication and secure group-chat runtime.
- **CrisisLinkForegroundService.**
Authorized responder telemetry synchronization with offline queue flush after validated internet recovery.
- **InternetCallForegroundService.**
WebRTC call runtime with microphone, phone-call, and media-projection foreground types (screen share), plus Telecom-framework integration for lock-screen ringing.
- **OfflineDownloadService.** Offline map region download lifecycle.

Additionally, a boot/Bluetooth-state receiver re-initializes listener readiness after restarts. Mesh/crisis-link auto-start behavior remains governed by explicit operator settings and authorization posture. Institutional deployments should validate this lifecycle on OEM-specific power policies because aggressive battery managers can alter expected background behavior.

13 Cloud Services: Trust Establishment, Telemetry, and Ciphertext Relay

Crisis Connect can be deployed in a fully offline configuration for contact-to-contact communication; the local incident path never depends on cloud availability. Cloud services now play three separately governable roles:

- **Trust establishment (setup-time).** Issuing responder role certificates and associated policies, distributing trust anchors for offline verification, and controlled distribution of responder credentials during an operation.
- **Operational telemetry.** Synchronizing responder operational telemetry (Crisis Link) and SOS-uplink observations without carrying emergency message content.
- **Ciphertext relay (internet path).** Routing end-to-end encrypted message envelopes and call signaling between internet-capable contacts, plus push wake-up delivery. The relay stores ciphertext only, deletes envelopes on acknowledgement with TTL purge, and is specified in Section 8. Earlier editions excluded cloud from message relay entirely; this edition replaces that rule with the narrower and testable invariant that *plaintext* emergency content never leaves end-user devices.

13.1 Scaling certificate distribution

For institutional deployments, certificate issuance should be integrated with an existing identity and access management (IAM) process (e.g., responder enrollment, device fleet provisioning, and periodic credential rotation). Operational scaling is primarily bounded by (i) enrollment throughput, (ii) device management capacity, and (iii) training and governance procedures.

13.2 Institution-controlled hosting option

Public authorities (e.g., AFAD) can operate the optional trust-establishment services on their own infrastructure and keep identity / operational data within institution-controlled environments. In this model, IAM can be integrated with a self-hosted Keycloak deployment (or an equivalent institutional IdP), while plaintext emergency content remains on end-user devices: local messaging continues to flow device-to-device, and internet-path messaging transits the relay only as end-to-end encrypted envelopes that institution- or vendor-operated infrastructure can route but not decrypt.

13.3 Crisis Link telemetry sync profile (v1.1.7)

When enabled by authorized responders, Crisis Link synchronization publishes rescue telemetry snapshots (for example active SOS signal observations, responder role / agency

context, and optional location snapshot) to institutional backend collections. This channel is operational telemetry only; it does not carry emergency chat payload content.

Operational safeguards in baseline:

- role-gated activation (local certificate posture required),
- queue-on-offline behavior with bounded pending payload retention,
- automatic flush when validated internet connectivity returns,
- schema-versioned writes for panel-side compatibility governance.

13.4 Revocation under offline constraints

Offline operation limits real-time revocation. In the validated v1.1.7 baseline, enforced controls are:

- short-lived role certificates (current issuer TTL profile: 72 hours),
- offline signature / owner / role / time-window checks against pinned trust anchor,
- mandatory online refresh when no usable cached certificate remains.

CRL / denylist snapshots and emergency revocation push workflows remain deployment-level extensions. For earthquake-scale operations with intermittent connectivity windows, authorities SHOULD predefine:

- who publishes signed denylist snapshots and what version cadence applies during brief connectivity windows,
- maximum accepted snapshot staleness before responder workflows are degraded or blocked,
- field SOP for forced re-verification after a missed snapshot refresh.

The selected revocation operating model MUST be explicitly documented by the operating authority and validated in exercises before broad deployment approval.

13.5 Deployment hardening requirements (production)

Production deployments MUST enforce strict separation between development assets and operational trust material, with controlled key management and auditable issuance workflows.

Minimum hardening requirements:

- Keep signing private keys outside source code (HSM/KMS/Secret Manager-backed issuance path).
- Version and rotate authority keys under documented policy.
- Bind certificate issuance to auditable role/identity governance (field-team/admin scopes).
- Publish key-rollover and emergency revocation procedures as operational runbooks.

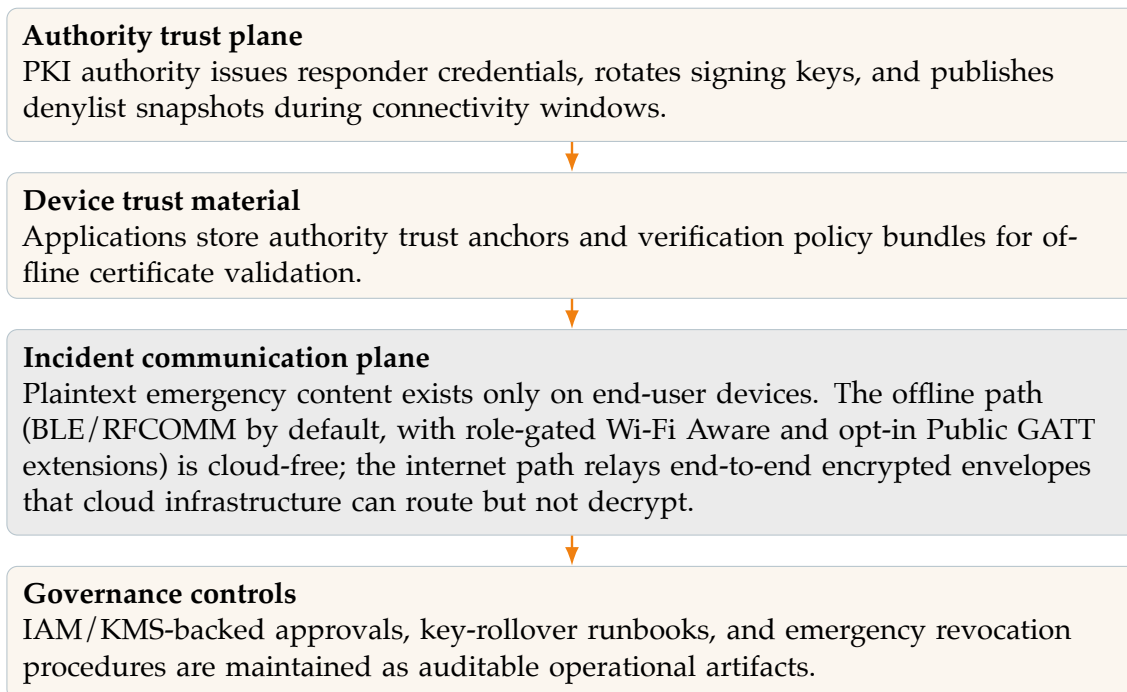


Figure 3: Cloud-assisted trust architecture with strict plaintext separation from cloud infrastructure (diagram version DGM-TRUST-2.0; legend: stacked layers represent governance and trust boundaries; the highlighted incident layer holds the only plaintext copies of emergency content).

Emergency message content on designated secure paths remains end-to-end encrypted between devices, while the opt-in Public GATT general channel is treated as non-confidential coordination traffic in the current baseline. Routine messaging does not depend on cloud availability: the internet drain accelerates and extends delivery when infrastructure survives, and its absence leaves local operation intact.

14 Web Dashboard and Self-Hosted Operations Architecture

The reviewed `crisis-connect-web` workspace is the authority operations-dashboard and control-plane line for institutional deployments. It is a Next.js 15.5.19 application targeting Node.js 20 with React 18.3.1, `next-intl`, Firebase client/admin SDKs, Drizzle/Postgres, `jose`, Leaflet, and a Docker/Helm on-prem packaging path. Its package version is 0.1.0; deployment version, image tag, and build SHA are separately reported at runtime through environment variables so institutional packets can bind a deployed image to a source tree, migration set, and build record.

This section treats the web application as a *control plane with an operator communication surface*. It can manage users, authority panels, identity-provider configuration, licensing, updates, health reporting, audit records, workflow settings, responder telemetry, and SOS signal observations; since July 2026 it also participates in the E2E messaging plane and group calls as an *endpoint* (Section 8). It is not a relay for the offline local incident path, and backend collections in the messaging plane hold ciphertext only. The professional deployment interpretation is therefore: use the web line for governance, visibility, identity, lifecycle management, and operator communications whose plaintext exists only at endpoints; keep the offline incident content plane local to the device transport layer.

14.1 Reviewed web baseline and release boundary

Reviewed item	Current statement
Workspace	Local <code>crisis-connect-web</code> source tree reviewed for dashboard, Settings, Crisis Sentinel provider / RAG / model-release governance, AI assistant, workflow builder / settings automation, Docker, Helm, on-prem mode, authentication / session handling, storage abstraction, licensing, update-management, and health-reporting behavior.
Version line	<code>package.json</code> version <code>0.1.0</code> ; runtime version metadata is read from <code>CRISIS_CONNECT_VERSION</code> , <code>CRISIS_CONNECT_IMAGE_TAG</code> , and <code>CRISIS_CONNECT_BUILD_SHA</code> .
Primary app stack	Next.js App Router, TypeScript, React, Tailwind, <code>next-intl</code> , Firebase, Drizzle / Postgres, Keycloak / OIDC, enterprise OIDC, and Leaflet map visualization.
Operational surface	Authority panels, responder and legacy-team telemetry, SOS signal observations, panel hierarchy, Settings administration, audit records, Crisis Sentinel providers, RAG knowledge, model-release metadata, workflow state, workflow settings, workflow credentials, SSO configuration, deployment license state, signed update posture, and installation-health metadata.
Release boundary	Local development artefacts such as <code>.env.local</code> , <code>.secrets</code> , logs, heap snapshots, <code>node_modules*</code> backups, and Playwright screenshots are not release-package material and MUST be excluded from published review bundles unless explicitly sanitized.
Incident boundary	Dashboard telemetry and administrative state are distinct from emergency message content transport. Local device-to-device message paths remain outside the dashboard.

14.2 July 2026 capability expansion

Between July 2 and July 16, 2026 the web workspace absorbed a large capability delta (on the order of five hundred changed files) that this edition documents as part of the reviewed baseline. The additions group into five planes:

- **Operator messaging and calls.** An E2E-encrypted messaging surface (attachments, voice notes, read receipts, typing indicators, forwarding, link previews) including cross-panel hierarchy messaging with server-issued channel keys held in server-only collections; 1:1 WebRTC calls; and multi-party SFU group calls with per-frame MLS media encryption (OpenMLS-based worker, transactional group-creator claim, safety numbers in the call UI). TURN moved to globally anycast infrastructure with short-lived credentials minted server-side. Cross-panel messaging is additionally governed by

a server-side `messagingPolicies` collection, editable only through admin/super-admin routes with the panel relationship verified server-side, so which panels may message which is an auditable policy decision rather than hard-coded behavior. Firestore rules were extended accordingly: key collections are server-only, call-room listing is deny-by-default with room ID as capability, and receipts/typing/presence writes are own-document-only.

- **External assistant channels (Telegram/Slack).** Field personnel can query the Crisis Sentinel assistant from Telegram or allowlisted Slack channels without a dashboard session. External channels get a deliberately reduced-privilege single-turn engine (“Sentinel Lite”): panel provider plus RAG knowledge recall, *no* tool belt, capped temperature and output. Authentication is timing-safe (hashed per-panel webhook secret for Telegram; HMAC request signing with a five-minute replay window for Slack), chat/channel allowlists gate participation, panel-scoped rate limits and body-size caps apply, secrets live in the AES-256-GCM workflow secret envelope, and channel enablement, reconfiguration, and disablement are audit-logged (Telegram allowlist-only edits are not currently logged — a noted follow-up item). The dashboard chat surface itself was consolidated onto a single streaming endpoint shared by web and mobile clients, with image-attachment (vision) support normalized across all configured AI providers and a refreshed, deprecation-pruned model catalog.
- **Workflow automation maturity.** A scheduler with POSIX-cron expressions (IANA-timezone wall clocks, DST-tested) and a queue mode whose transactional next-run claim acts as the deduplication lock, so multiple replicas add throughput without double-firing; retry queue with dead-lettering, durable suspension resume/expiry, error-workflow hooks, per-panel heartbeats, and audit records under a system-scheduler actor. Suspended runs can resume through signed public callback links (HMAC capability tokens, fail-closed without the signing secret). A workflow can also be exposed as an MCP tool over a bearer-authenticated JSON-RPC endpoint so external AI agents can invoke crisis workflows under audit. New node families include panel-scoped data tables, a read-only data-store query node that exposes panel operational data to flows while blocking secret/credential collections and path traversal, a deliberately model-free AI guardrails screen (prompt-injection heuristics, PII redaction, domain allowlists), AI evaluation runs over dataset tables with optional LLM-judge scoring, structured-output/reranker/sentiment utilities, a respond-to-webhook node with configurable response modes, typed connector operations, and RSS/SSE/file triggers. Branch execution remains sequential by default (parallel execution is opt-in and forced back to sequential around merge/loop/wait nodes); cron activation schedules the next wall-clock occurrence rather than firing immediately, and an unparseable stored expression degrades to interval polling rather than stalling. Panel MCP bearer tokens are generated, rotated, and revoked from the workflow editor, displayed exactly once at creation, and gated by the publish-tier permission; their lifecycle should fall under the same rotation and audit expectations as webhook secrets.
- **Security and platform hardening.** Runtime secret resolution moved to a managed secret store for TURN and callback keys (fixing a redeploy-drops-secrets failure class); de-

pendency advisories were closed on the websocket and spreadsheet libraries (the latter by sourcing the maintained SheetJS build from the vendor's official distribution channel, because the npm-registry package is frozen with unfixed advisories — a supply-chain provenance detail deployment packets should record); phone verification and SMS login landed with an App Check conflict fix; every new public surface (assistant webhooks, MCP, public callbacks) carries the shared panel rate-limit bucket and fail-closed secret gates; and all outbound HTTP performed on behalf of external assistant channels and workflow code/HTTP nodes is routed through a shared SSRF-guarded fetch layer. Playwright end-to-end test infrastructure now exists (workflow suites in memory mode and against a Keycloak/SSO server), alongside roughly a hundred new or changed test files and a `workflow:preflight` deployment-preflight script that should join the acceptance evidence packet.

- **Deployment and operations.** Docker Compose gains opt-in profiles for the workflow scheduler and installation-health reporter plus a scripted Keycloak bootstrap; the Helm chart adds CronJobs for the scheduler (per-minute), license renewal (twelve-hourly), and health reporting (five-minutely), with all new secrets injected through the chart's secret template. Signed ES256/JWS update manifests remain the update-governance mechanism.

Public positioning and disclosure. The public site was repositioned from offline-only to the hybrid story (E2E internet messaging and calls when online, direct Bluetooth when networks fail); a trust-center page now publishes verification resources, security practices, and a subprocessor table — currently naming Firebase/Google Cloud (region `us-central1`), Cloudflare (SFU/TURN), MapTiler and CARTO (map tiles), and the mobile app stores, which institutional reviewers should treat as the current external-dependency inventory for the cloud-hosted plane, including its United States data-residency default — and explicitly discloses what the project does *not* yet have: an independent security audit or penetration-test report, ISO 27001 or SOC 2 certification, and a paid bug bounty. A Turkish data-protection (KVKK) disclosure page exists in source but is feature-flagged off pending legal review and **MUST NOT** be cited as published. These self-declared gaps are carried into the open-gaps register below.

Versioning discipline note. The package version remains `0.1.0` with no git tags or changelog; release identity is carried operationally by `CRISIS_CONNECT_VERSION`, image tag, build SHA, and signed update manifests. Institutional packets **MUST** therefore bind review claims to image tag and build SHA rather than to the static package version.

14.3 Operational data classification and authority use

For institutional review, the web dashboard should be assessed by data class rather than by screen count. The relevant distinction is whether a record supports administration and operational awareness, or whether it would become incident message content. The reviewed web baseline belongs to the former category.

Data class	Examples in the web line	Operational treatment	Boundary statement
Administrative configuration	Users, roles, panel metadata, SSO connection records, license tokens, update settings.	Governed by authority IAM, audit logging, secret-management policy, and change-control procedures.	Administrative state; not incident content.
Operational telemetry	Responder last-seen state, device/team identifiers, panel signal summaries, SOS observation windows, map markers.	Used for situational awareness and dispatch support; retention and access should be policy-defined.	Metadata and observations; not a guaranteed casualty count or message transcript.
Audit and lifecycle records	Session-start audit events, workflow execution state, license status, update manifest status, health heartbeat metadata.	Used for accountability, support, and release governance; should be immutable or append-only where policy requires.	Governance evidence; not a user-message relay.
Ciphertext messaging state	Sealed E2E envelopes, encrypted channel messages and attachments, opaque MLS handshake blobs, call-signaling documents.	Server-side collections hold ciphertext only; keys are issued to authorized members and key collections are server-only; retention follows acknowledgement-deletion and TTL policy.	Encrypted operator communications; plaintext exists only at endpoints.
Excluded incident payloads	Plaintext crisis chat text, local secure-chat payloads, local voice payloads, and offline device-to-device message bodies.	Remain on end-user devices and local transports; never routed through the dashboard in plaintext.	Explicitly outside the reviewed web control plane.

14.4 Runtime modes and storage abstraction

The web application supports three runtime modes selected by `DEPLOYMENT_MODE`. The browser bundle also needs `NEXT_PUBLIC_DEPLOYMENT_MODE` set at build time so the client chooses the correct dashboard behavior. This is a material deployment control: a self-hosted review must verify both build-time and run-time mode selection, because a server configured for on-prem mode with a browser bundle built for cloud behavior would produce inconsistent dashboard behavior.

Mode	Storage	Auth	Intended use
cloud	Firestore through Firebase Admin/server drivers and Firebase client realtime listeners.	Firebase Auth, with optional enterprise OIDC bridge.	SaaS, small municipalities, NGOs, public website hosting, and managed Firebase deployments.
onprem	Postgres through a generic StorageDriver and path router that maps Firestore-shaped paths to relational tables.	Keycloak OIDC and/or enterprise OIDC; app-issued httpOnly session cookies.	National agencies, regulated customers, air-gapped-adjacent environments, and authority-owned infrastructure.
memory	In-memory driver.	In-memory / test provider.	Automated tests only; not a deployment mode.

The key architectural choice is that server routes use `getStorageDriver()` instead of directly assuming Firestore. In cloud mode, the driver wraps Firestore. In on-prem mode, the driver uses Postgres and a path router. The router maps logical document paths such as `agencyPanels/{panelId}/responders/{responderId}` to concrete Postgres handlers and tables. This preserves the dashboard's document-oriented product model while allowing a self-hosted relational backend. It also creates a clear review surface: each authority workflow should be traceable either to a storage-driver-backed server API or to an explicitly retained cloud-mode client listener.

Logical path	Postgres table	Operational purpose
<code>users/{uid}</code>	<code>users</code>	Dashboard users, role/profile data, panel binding, and permission flags.
<code>ssoConnections/{providerId}</code>	<code>sso_connections</code>	Enterprise OIDC providers, allowed domains, role mappings, JIT policy, and client-secret references.
<code>agencyPanels/{panelId}</code>	<code>agency_panels</code>	Authority panel summary, hierarchy fields, dashboard counters, and agency metadata.
<code>agencyPanels/{panelId}/responders/{id}</code>	<code>panel_responders</code>	Verified field-team/responder stream, device IDs, user IDs, agency binding, and latest telemetry payload.
<code>agencyPanels/{panelId}/signals/{signalId}</code>	<code>panel_signals</code>	SOS signal observations, active/resolved state, reporter linkage, GPS/victim-location payloads, and last-seen timestamps.
<code>agencyPanels/{panelId}/signals/{signalId}/reporters/{uid}</code>	<code>signal_reporters</code>	Per-responder signal reporting lineage and reporter-side metadata.

Logical path	Postgres table	Operational purpose
agencyPanels/{panelId}/roles/{roleId}	panel_roles	Panel-level access role catalog and permission maps.
agencyPanels/{panelId}/orgNodes/{nodeId}	panel_org_nodes	Organization hierarchy nodes, parent links, role bindings, and member mapping.
agencyPanels/{panelId}/config/{docId}	panel_config	Panel settings, admin profile, license token storage, and operational configuration.
agencyPanels/{panelId}/auditLogs/{logId}	panel_audit_logs	Immutable server-written audit trail for settings/session/workflow events.
agencyPanels/{panelId}/workflows/*	workflow tables	Workflow drafts, versions, executions, defaults, providers, credentials, idempotency, and rate-limit state.
rag_chunks	rag_chunks	pgvector-backed knowledge chunks for Sentinel/RAG workflows; intentionally queried directly rather than through the generic document router.

14.5 Dashboard session and access flow

Protected dashboard routes use two layers of access control. Middleware blocks unauthenticated browser navigation to `/dashboard`, `/settings`, and `/super-admin` if no structurally valid `__session` cookie is present. This middleware gate is an early routing control, not the final authorization decision. The route layout then performs the authoritative server-side check by verifying the session cookie, loading `users/{uid}`, and applying dashboard access rules.

1. The browser signs in through Firebase Auth, Keycloak OIDC, or enterprise OIDC.
2. The app creates an `httpOnly __session` cookie through `/api/auth/session`. The current session lifetime is 12 hours.
3. The session route verifies an ID token with the active auth provider, writes a dashboard-session audit log when the user has dashboard access, and clears legacy client cookies.
4. `/api/auth/me` resolves the server session and returns the authenticated dashboard payload for client state.
5. `/api/dashboard/overview` checks the same session context, resolves authorized panel scope, enforces the dashboard license gate, and returns no-store panel data.

Dashboard access is role and flag based. Accepted roles are admin, authority, and field-team variants. A `dashboardAccess` flag may explicitly disable access. Platform/global users can enumerate panels; ordinary users are constrained to their bound panel, child panels, and configured ancestors. The review-relevant point is that dashboard authorization is enforced server-side before protected panel data is returned; client-side state is not the trust anchor.

Access-control layer	Reviewed behavior	Professional review expectation
Routing gate	Middleware checks for a structurally valid <code>__session</code> cookie before entering protected dashboard/settings routes.	Treat as a fast navigation guard only; do not rely on it as the final authorization control.
Server session verification	Layout and APIs verify the session through the active provider and server-session logic before returning dashboard state.	Treat server verification as the authoritative dashboard access control.
User and panel authorization	User profile, role, dashboard access flag, panel binding, hierarchy, and global/platform flags constrain panel access.	Reviewer packet should include role-to-panel test cases for ordinary, authority, and platform/global users.
License feature gate	Dashboard overview data is gated by licensed dashboard feature checks.	Pilot packets should declare license mode (<code>observe</code> or <code>strict</code>) and expected failure behavior.

14.6 Dashboard data flow and operational UI

The dashboard page exposes a live operations view: panel status cards, responder/team lists, SOS signal lists, alert-window filtering, status filters, panel hierarchy selector, and a Leaflet map. It derives active-vs-idle responder state from telemetry recency, merges responder and legacy team sources, and treats active signals as time-windowed observations rather than permanent victim truth.

14.7 Settings administration surface

The Settings route is not a single simple preferences page. In the reviewed source it is the primary administrative surface for account controls, notification preferences, panel administration, operator and user management, permissions, role scope, license state, on-prem update visibility, security logs, Crisis Sentinel provider configuration, and rescue certificates. That breadth makes Settings one of the most important areas to distinguish between already server-backed controls and remaining client/Firebase-heavy controls.

Dashboard behavior	Current implementation	Review implication
Panel overview API	<code>/api/dashboard/overview</code> returns panel options, selected panel summary, responders, legacy teams, and signals. Limits are 250 responders, 250 teams, and 1000 signals.	On-prem dashboards can be loaded from server APIs backed by Postgres; responses are <code>no-store</code> .
Cloud realtime listeners	Cloud mode still uses Firestore client listeners for panels, responders, teams, and signals.	Firebase rules remain part of the cloud security boundary; cloud dashboard has richer realtime behavior.
On-prem dashboard path	In server/on-prem mode the page loads the overview API instead of relying on Firebase client listeners.	Main dashboard is on-prem aware, but deep client-heavy screens still need migration before claiming a fully Firebase-free admin product.
Signal recency	Active signal window is 2 minutes; dashboard alert windows include 1 h, 3 h, 5 h, 24 h, 48 h, 10 d, and 30 d views.	Operational counts are observation windows, not verified casualty totals.
Responder recency	Field team idle threshold is 3 minutes unless a status is explicitly provided.	Last-seen behavior must be explained during pilots so stale telemetry is not treated as real-time location certainty.
Map dependency	The current map component uses external Carto basemap tiles and Leaflet markers for teams and SOS signals.	Fully isolated deployments should replace public tile URLs with an agency-hosted map tile service or another approved map source.

Settings area	Reviewed implementation pattern	Professional assessment
Settings shell and route guard	<code>app/[locale]/settings/layout.tsx</code> verifies the server session and user document before protected access; <code>app/[locale]/settings/page.tsx</code> renders a client tab shell.	The page is protected by server-side session logic, but much of the tab content remains client-rendered. Server guard should be treated as necessary but not sufficient proof that every tab is on-prem complete.

Settings area	Reviewed implementation pattern	Professional assessment
Tab and feature model	The shell conditionally exposes general, account, notifications, admin, operators, users, permissions, role scope, license, updates, security, Crisis Sentinel, and certificates. The on-prem update tab depends on the public deployment-mode flag being set to <code>onprem</code> and license-management permission.	This is a mature administrative taxonomy. Reviewers should ask which tabs are required for a given authority pilot and then verify the backing data path for each required tab.
Account and notification settings	<code>LiveSettingsSections.tsx</code> uses Firebase Auth operations and Firestore client writes/listeners for profile, password, phone, MFA-like account flows, avatar/profile state, notification preferences, and account deletion requests.	These flows are not yet a clean Postgres-only/on-prem story. They need storage-backed account/profile APIs or a Keycloak/enterprise-IdP-specific implementation before they can be claimed as Firebase-free.
Panel admin, operator, roles, and scope	The same live settings bundle listens to and writes panel organization nodes, roles, admin profile/config, scoped themes, and related hierarchy data through Firestore client APIs.	This is operationally valuable, but the current client-listener pattern should be migrated into server APIs that use the storage driver and enforce panel/role authorization consistently in on-prem mode.
User account management	<code>/api/admin/users</code> lists, creates, and updates managed users through the active auth provider and the storage abstraction. It checks manager role, panel binding, dashboard access, license feature and capacity, and writes audit logs.	This is one of the stronger Settings sub-surfaces for on-prem readiness because it is API-backed, driver-aware, license-aware, and audit-aware. It should be included in the positive self-hosted evidence packet.
License and update controls	<code>/api/license/status</code> , <code>/api/license/renew</code> , and <code>/api/onprem/update/status</code> are server-backed. The update panel displays signed-manifest status and Docker/Helm commands but does not execute remote updates.	This is the correct governance pattern: verification and operator guidance are inside the app, while actual upgrades remain authority-controlled change-management actions.

Settings area	Reviewed implementation pattern	Professional assessment
Security logs and audit policy	Client audit writes are routed through <code>/api/audit-logs</code> ; purge and alerting controls use server routes. The security tab still reads the audit stream through client-side listeners.	Audit write integrity has moved server-side, which is positive. For fully self-hosted operation, security-log reads and retention views should also have storage-backed server read APIs.
Certificates	The certificates surface uses <code>/api/certificates</code> and revoke routes backed by server authorization and audit logging.	This is an authority-control feature rather than a message transport feature; reviewer packets should show who can view and revoke rescue-role certificates.
Crisis Sentinel provider settings	<code>/api/admin/crisis-sentinel</code> exposes provider configuration, built-in quota usage, per-provider model defaults, base URLs, host allowlists, private-network flags, timeout/output settings, and monthly caps. Writes require <code>admin/authority-style</code> edit permission and the <code>crisis_sentinel</code> license feature.	This is a materially important AI governance surface. It supports managed providers and OpenAI-compatible/self-hosted providers, but production deployment must keep provider API keys, private-network access, and model-selection authority under institutional control.
Crisis Sentinel model and mobile-release controls	<code>/api/admin/crisis-sentinel/models</code> performs dynamic model discovery with bounded response reads and secret redaction. <code>/api/admin/crisis-sentinel/mobile-release</code> validates release identifiers, HTTPS locations, SHA-256 hashes, byte sizes, platform keys, and approved model-storage paths.	These routes show real operational hardening, especially around model listing and mobile model artifact metadata. The remaining production expectation is secret storage, model artifact provenance, and storage-policy evidence.

The defensible Settings claim is therefore nuanced: user management, licensing, on-prem update status, certificates, audit writes, audit retention controls, and Crisis Sentinel provider configuration already have meaningful server/API governance. Account/profile, notification, organization-role live editing, and some security-log reads still depend on Firebase client behavior and should be listed as migration items in a self-hosted production packet.

14.8 Crisis Sentinel governance and model lifecycle

Crisis Sentinel is the AI decision-support layer inside the web control plane. It is not a separate message transport network and it is not an automatic incident commander. In the reviewed implementation, it combines provider configuration, model discovery, built-in quota metering, panel-scoped RAG, multi-agent incident assessment, workflow generation, workflow execution support, and mobile model-release metadata. That makes it a high-value governance surface: the important question is not merely whether the product can call an AI model, but who may configure the model, where prompts and knowledge are stored, which tools the agent may call, and how model artefacts are released.

Crisis Sentinel control	Reviewed implementation	Operational interpretation
Provider families	The provider registry supports Gemini, OpenAI, Anthropic, Mistral, Groq, OpenRouter, and OpenAI-compatible/self-hosted endpoints. Panel providers are resolved before the built-in Gemini fallback.	Authorities can either use a platform-provided built-in key with quotas or bring their own provider. Procurement materials should name the selected provider and data-residency posture.
Provider security controls	Provider configs live under panel-scoped storage paths. API keys are encrypted/masked; default model, temperature, output-token cap, timeout, monthly request/token caps, base URL, private-network allowance, and host allowlist are configurable where applicable.	AI provider setup is an administrative security decision. Private-network access and OpenAI-compatible base URLs should require explicit authority approval and network allowlisting.
Model discovery and testing	Model-listing routes call native or OpenAI-compatible model APIs with bounded response reads and secret redaction. Provider test routes overlay unsaved editor fields on stored config and enforce the provider-specific constraints before returning a test result.	Operators can validate keys/models without exposing raw secrets in the reviewer packet. Failed upstream model listing should be documented as an integration state, not hidden from acceptance review.
Built-in usage limits	Built-in provider usage is metered per panel across day, week, and month buckets. The quota reservation path rejects calls when request or token caps are exceeded. Panel-owned providers may also have monthly caps.	Quota behavior must be part of pilot planning. A missing or exhausted provider should degrade predictably instead of silently changing models or bypassing policy.

Crisis Sentinel control	Reviewed implementation	Operational interpretation
Knowledge/RAG ingestion	<code>/api/admin/crisis-sentinel/knowledge</code> lets authorized admins ingest or delete panel-scoped runbooks and past-incident summaries. The route limits documents and text size, chunks and embeds content, and writes to the deployment-mode vector store. Cloud uses Firestore vector search, on-prem uses pgvector, and tests use memory.	RAG content is operational doctrine. It should be approved, versioned, and retained like a runbook, because it shapes every grounded AI answer.
RAG retrieval and citations	The retriever embeds natural-language queries, returns panel-scoped chunks, and formats citation-tagged context. The Knowledge Officer sub-agent uses this path to cite panel runbooks or incident precedent.	Crisis Sentinel can be grounded in agency knowledge instead of generic model memory, but reviewers should see source IDs and citation behavior.
Multi-agent assessment	The Crisis Commander orchestration runs specialist sub-agents: Triage Officer, Situation Analyst, Knowledge Officer, and Dispatch Planner. It supports parallel specialist fan-out and manager/delegate-style orchestration. Chat can call <code>run_crisis_assessment</code> for new or high-severity incidents.	This is decision support, not automatic command. The output should be treated as an auditable incident brief requiring human command review.
Workflow generation boundary	The assistant's <code>createWorkflow</code> tool and <code>/api/workflows/ai/generate</code> build reusable workflow drafts. The route requires <code>workflow:create</code> , provider availability, quota reservation, prompt limits, and parsed workflow output.	Workflow generation is a drafting capability. Generated graphs still need operator review, credentials, permission checks, dry-run, and publication before live use.
Mobile model-release metadata	The mobile-release route validates platform key, release ID, display/file names, HTTPS or approved storage location, SHA-256 hash, byte size, and minimum free bytes. Approved Firebase Storage buckets and the <code>crisis-sentinel/models/</code> prefix are enforced.	Edge/mobile model artefacts need provenance evidence: hash, expected size, approved object path, release owner, and rollback plan.

The strongest Crisis Sentinel statement is therefore: the web line already contains meaningful AI governance primitives, including provider control, quotas, RAG knowledge ingestion, multi-agent assessment, workflow drafting, and model-release metadata. The production statement should still require authority-approved provider selection,

prompt/knowledge retention policy, model artefact provenance, MCP/tool allowlisting, and explicit human approval before AI outputs become operational actions.

14.9 AI assistant and chat surface

The `/ai` page is the browser-based Crisis Sentinel assistant surface. Its server page mainly provides metadata and delegates the product experience to `AiPageClient.tsx`. The client manages chat sessions, message rendering, attachments, voice/dictation UI state, library/search behavior, workflow preview cards, response navigation, and dashboard context handoff. It can read `dashboard_context` from `sessionStorage`, so operators can move from the dashboard into the assistant with agency/panel context already available.

AI surface element	Reviewed behavior	Deployment implication
Chat storage	The client reads and writes <code>agencyPanels/{panelId}/chats</code> and nested <code>messages</code> through Firestore client APIs, including realtime listeners and message-cache search.	The AI chat history path is currently cloud/client-oriented. A Firebase-free on-prem claim requires server-backed chat repositories or a Postgres-backed realtime/subscription replacement.
Assistant request path	User messages are sent to <code>/api/chat/stream</code> , the single consolidated streaming chat endpoint shared by web and mobile clients (the legacy non-streaming <code>/api/chat</code> route was removed in July 2026); the streamed response may include text, tool-call metadata, and generated workflow hints.	The assistant should be described as an operator decision-support surface, not an emergency message relay. Its prompts, logs, attachments, and outputs need separate retention and redaction policy.
Dashboard context	The AI client can hydrate context from <code>sessionStorage</code> values created by dashboard navigation.	This is useful for operator workflow continuity, but it also means the assistant should inherit the same panel-scope and data-minimization expectations as the dashboard.
Inline workflow generation	When the assistant receives a <code>createWorkflow</code> tool call, it stores a <code>workflowGen</code> state on the bot message, calls the AI workflow generator, streams a graph preview, and lets the operator apply the generated workflow into the workflow builder through a <code>sessionStorage</code> handoff.	This makes AI useful for workflow drafting, but the operator still needs review, permission checks, and credential completion before execution. Generated workflows should not be treated as automatically approved operational automations.

AI surface element	Reviewed behavior	Deployment implication
Current self-hosted status	The AI page imports Firebase client Firestore and maintains chat UI state in browser-local storage plus Firestore.	The correct current statement is that AI assistant UX exists and is integrated with dashboard/workflows, but its chat persistence must be migrated before the complete AI assistant can be called on-prem independent.

Beyond these surface elements, the assistant's operational reach is defined by an explicit, reviewable tool belt rather than open-ended model behavior. The reviewed July 2026 line declares **38 chat-runtime functions** in a single canonical catalog (`lib/ai/tool-catalog.ts`), kept in sync with the function declarations of the consolidated `/api/chat/stream` endpoint, and surfaces them to operators on a dedicated `/ai/tools` catalog page with card and per-tool usage views. The inventory below is the assistant's complete AI capability surface as reviewed; it is the concrete referent for the governance, threat-model, and data-residency language in this document.

Assistant tool	Reviewed function	Exposure and effect class
Panel telemetry and governance widgets		
<code>showLocationOnMap</code>	Plots team members or SOS signals as an interactive map widget inside the chat timeline.	All operators; chat widget, read-only render.
<code>showSosSignals</code>	Renders the panel's active SOS/distress signals; the widget pulls live dashboard data rather than model-supplied values.	All operators; chat widget, read-only render.
<code>getTeamStatus</code>	Renders live field-team status and headcount; the widget pulls live dashboard data rather than model-supplied values.	All operators; chat widget, read-only render.
<code>showOpsStatus</code>	Live operational health: workflow run counts and failures, scheduler heartbeat state, retry queue, open operational alerts.	Manager roles; server-executed read.
<code>showLicenseStatus</code>	Deployment license validity, expiry countdown, licensed features and limits.	Manager roles; server-executed read.
<code>showAiUsageMetrics</code>	Per-tool AI call counts and last-used times plus workflow AI token/cost totals for the panel.	Manager roles; server-executed read.
<code>showSecurityLogs</code>	Renders the security audit-log widget with optional type, search, and limit filters.	Manager roles; chat widget, read-only render.
<code>showPermissionsAndRoles</code>	Renders the roles/permissions matrix or organization-tree widget with optional filters.	Manager roles; chat widget, read-only render.
Knowledge, recall, and web search		
<code>searchKnowledge</code>	Retrieval over the panel's own incident/runbook knowledge base (RAG); results are cited to ground answers.	All operators; server-executed read.

Assistant tool	Reviewed function	Exposure and effect class
searchChatHistory	Searches only the operator's own past assistant chats (titles and messages) to recall earlier decisions.	All operators; server-executed read.
getSituationReports	Searches verified humanitarian situation reports on ReliefWeb (UN OCHA); returns titles with links.	All operators; server-executed read.
searchWeb	Public web search. On Gemini 3+ the provider-native search tool is used; on Gemini 2.x the same capability is exposed as a server-executed function so custom tools and web search can coexist.	All operators; server-executed read.
External hazard, weather, and geospatial data (keyless public sources)		
getWeather	Live conditions (temperature, feels-like, humidity, wind, rain) for a place or coordinates via Open-Meteo.	All operators; server-executed read.
getAirQuality	Live air quality (US AQI, particulate and gas readings) via Open-Meteo, for wildfire smoke, dust, or chemical-incident air hazards.	All operators; server-executed read.
getWeatherAlerts	Official severe-weather warnings: US locations via NWS, European countries via MeteoAlarm; the tool itself documents that Türkiye (MGM) is not covered.	All operators; server-executed read.
getHazardEvents	Active wildfire, flood, severe-storm, volcano, and landslide events from NASA EONET, optionally focused near a place.	All operators; server-executed read.
getFloodRisk	31-day river-discharge / flood-risk forecast (Copernicus GloFAS via Open-Meteo).	All operators; server-executed read.
getDisasterAlerts	Current GDACS (UN/EC) disaster alerts with Red/Orange/Green humanitarian impact levels.	All operators; server-executed read.
getEarthquakeFeed	Recent earthquakes from the panel-configured seismic source: USGS worldwide, AFAD for Türkiye, or USGS/FEMA for the US.	All operators; server-executed read.
getEarthquakeImpact	USGS PAGER impact assessment: alert level, maximum shaking intensity, felt reports, and tsunami flag.	All operators; server-executed read.
getMarineConditions	Live sea state (wave height/direction/period, swell, sea-surface temperature) via Open-Meteo Marine, for coastal SAR and shoreline flooding.	All operators; server-executed read.
getSatelliteImage	NASA Worldview / VIIRS satellite snapshot of an area (true-color or fire/flood false-color contrast) rendered in chat.	All operators; server-executed read.

Assistant tool	Reviewed function	Exposure and effect class
getRoute	Distance, ETA, and turn-by-turn steps between two places via OpenStreetMap/OSRM, for dispatch and evacuation questions.	All operators; server-executed read.
findNearestFacility	Nearest hospitals, clinics, pharmacies, shelters, fire stations, or police via OpenStreetMap.	All operators; server-executed read.
queryGis	Read-only query of the panel's configured ArcGIS feature layer (road closures, flood zones, infrastructure) with an optional SQL-style filter.	All operators; server-executed read against a panel-configured GIS credential.
Analysis and drafting		
assessDamageFromImage	Scores operator-attached damage photos (0–10 severity, hazards, summary) in field, satellite, or before/after change-detection mode.	All operators; server-executed analysis of operator-supplied images.
generateSitrep	Compiles a downloadable situation-report PDF from live panel data (active SOS, team status, recent events).	All operators; chat widget producing a client-side download.
createWorkflow	Generates a draft workflow graph from a natural-language description and streams a preview; the operator applies it into the workflow builder.	Requires the workflow-plane workflow:create permission; drafting only, no execution.
run_crisis_assessment	Runs the multi-agent incident assessment (scenario and parallel/manager coordination mode selectable) before answering.	Declared only when a Gemini runtime is resolvable; server-executed read.
Confirm-gated panel actions (manager-declared)		
broadcastAlert	Drafts a team alert for the panel's configured Slack/Discord messaging connector.	Manager roles; confirmation card, then /api/ai/broadcast .
logToSheet	Appends a server-timestamped row to the panel's configured Google Sheets log.	Manager roles; confirmation card, then /api/ai/sheet-log .
addKnowledge	Saves a runbook or incident note into the panel RAG knowledge base, shaping future grounded answers for every operator.	Manager roles; confirmation card, then the Crisis Sentinel knowledge API.
createOperatorAccount	Registers a new operator/field-team account; the one-time temporary password is generated server-side with a CSPRNG and never persisted into chat history.	Manager roles; confirmation card, then /api/admin/users .
updateAdminSettings	Updates the agency profile (authority name, region code, headquarters, organization template).	Manager roles; confirmation card, then currently a Firestore client write (migration item).

Assistant tool	Reviewed function	Exposure and effect class
Operator personal-account integrations		
sendEmail	Sends an email from the operator's own connected Gmail account.	All operators; confirmation card, then the operator's own connected account.
addCalendarEvent	Creates an event in the operator's own connected Google Calendar.	All operators; confirmation card, then the operator's own connected account.
sendSlackMessage	Sends a message from the operator's own connected Slack account.	All operators; confirmation card, then the operator's own connected account.
searchDriveFiles	Read-only file-name search over the operator's own connected Google Drive; returns names and links.	All operators; server-executed read over the operator's own connected account.

Four mechanisms make this tool belt governable, and all are enforced in the reviewed request path rather than in prompt text alone. **Role-scoped declaration:** the manager-only rows above are added to the model's toolset only when the authenticated user resolves to an admin/authority role; `createWorkflow` additionally requires the workflow-plane `workflow:create` permission, and `run_crisis_assessment` is offered only when a Gemini runtime is resolvable through the panel's own provider key or the platform key. **Per-panel disablement:** panel admins can disable any tool from the `/ai/tools` page through `/api/admin/ai-tools`; read access is open to dashboard users so the catalog and usage views render, writes require admin/authority settings-management rights, and the endpoint is license-gated on the `crisis_sentinel` feature and storage-driver-backed, so it works identically in cloud and on-prem modes. The chat runtime enforces the disabled list at request time twice: disabled declarations are removed before the model call, and any disabled call the model still emits is dropped with a user-visible notice. The manager-only server tools are additionally re-checked on both provider dispatch paths so a non-manager model call can never execute them. **Side-effect containment:** every server-executed tool in the inventory is a read-only lookup or render. Every state-changing action instead returns a pending confirmation card and executes only after explicit operator approval — `createOperatorAccount` through `/api/admin/users`, `broadcastAlert` through `/api/ai/broadcast`, `logToSheet` through `/api/ai/sheet-log`, and `addKnowledge` through the Crisis Sentinel knowledge API. One exception is reviewer-relevant: the approved `updateAdminSettings` action still writes its panel profile document through the Firestore client SDK rather than a server API, so it belongs on the same UI data-path migration list flagged elsewhere in this section. **Usage and quota accountability:** chat requests reserve panel-scoped provider quota before the model call, each executed tool increments per-tool, per-panel usage counters (count and last-used time) that back the catalog's usage view and the `showAiUsageMetrics` widget, and the earthquake-feed source region (USGS global, AFAD for Türkiye, USGS/FEMA for the US) is a panel-level setting rather than a per-request model choice.

Three review expectations follow from this inventory. First, tool enable/disable and hazard-region changes are not currently written to the audit log (unlike workflow settings changes); pilot packets should treat this as a follow-up item alongside the Telegram allowlist-edit gap noted earlier. Second, the confirmation gate is a client-mediated flow, so acceptance evidence should include negative tests proving that the server APIs behind each confirm-gated action independently enforce role and panel authorization when called outside the chat UI. Third, the external-data rows define the assistant’s outbound egress inventory (Open-Meteo, NASA EONET/Worldview, USGS, AFAD, GDACS, ReliefWeb, OpenStreetMap/OSRM, US NWS, MeteoAlarm); self-hosted deployments that restrict egress should allowlist exactly this set, and operators should note the documented coverage gap that MeteoAlarm carries no Türkiye warnings. Finally, the contrast with the external assistant channels is deliberate and should be preserved in deployment language: Telegram/Slack “Sentinel Lite” sessions receive none of these tools, so the full belt above exists only behind an authenticated dashboard session.

For institutional documentation, the AI assistant should be positioned as *governed operator support*: it can summarize, guide, and draft operational workflows, but it does not replace command approval, does not carry device-to-device emergency chat, and must be covered by AI provider, prompt-retention, audit, and data-residency policy.

14.10 AI workflow automation plane

The workflow subsystem is more advanced than a visual demo canvas. It combines a workflow builder UI, generated workflows, role-based permissions, provider/model settings, credentials, dry-run/live execution, versioning, webhook triggers, AI agent streaming, MCP tool references, memory, and audit/execution logging. The current implementation is mixed: several sensitive operations are already routed through server APIs and the storage abstraction, while the main workflow list/runs UI still uses Firestore client realtime listeners for parts of the experience.

Workflow control	Reviewed implementation	Governance meaning
Workflow UI	/ai/workflows renders WorkflowsClient.tsx with workflow cards, editor launch, import/export, credential view, run history, dry-run simulation, and AI handoff from the assistant.	The operator experience is feature-rich, but some list, chat, workflow, and execution reads still use Firestore client listeners. This should be listed as a UI data-path migration item for on-prem pilots.
Settings route authorization	/ai/workflows/settings is server-rendered first: it verifies the server session, loads users/{uid} through getStorageDriver(), resolves panel ID, asserts panel access, computes workflow role, and passes allowed actions to the client.	This is a strong governance pattern because workflow settings do not rely solely on browser state. It should be preserved for all workflow admin pages.

Workflow control	Reviewed implementation	Governance meaning
Workflow roles and actions	Roles are viewer, editor, operator, and admin. Actions are separated across view, create, update, delete, publish, execute, execution view, webhook secret management, memory browse/clear, credential view/create/update/delete, model view/update, permission management, and settings update.	The permissions model is reviewable and can support least-privilege operations. Evidence should include denial tests for editor-vs-operator-vs-admin behavior.
CRUD, publish, versions	<code>/api/workflows/[workflowId]</code> handles <code>get/update/create/delete</code> -style workflow operations. Versioning routes publish drafts, list versions, load versions, and restore previous versions through storage-backed helpers.	This supports change control: draft, publish, restore, and execution-source behavior can be audited instead of treating workflows as unversioned scripts.
Execution API	<code>/api/workflows/[workflowId]/execute</code> requires <code>workflow:execute</code> , supports dry-run and live modes, loads draft or published versions as appropriate, applies panel defaults, truncates results, persists execution records when settings allow, updates workflow stats, and writes audit logs atomically through the storage driver.	Execution is not just a browser action; it is a permissioned server operation. Production packets should include dry-run/live execution evidence and audit-log examples.
AI workflow generator	<code>/api/workflows/ai/generate</code> requires <code>workflow:create</code> , resolves the panel's active provider, supports built-in Gemini fallback, applies request/token quota reservations, sends bounded prompts, parses generated workflow JSON, and streams workflow nodes/edges through SSE.	AI generation is appropriately constrained as a drafting tool. It should be documented as assisted creation, not unattended automation approval.
AI agent stream and MCP tools	<code>/api/workflows/agent/stream</code> requires <code>workflow:execute</code> , loads approved/default crisis tools, resolves MCP server references, applies tool policy, optionally attaches workflow memory, and streams agent events.	This is the highest-risk AI automation surface because it can call tools. Production deployments need allowlists, credential scoping, MCP endpoint approval, memory retention limits, and audit visibility.

Workflow control	Reviewed implementation	Governance meaning
Credentials and providers	Workflow credentials are stored under panel-scoped <code>workflowCredentials</code> via the storage abstraction. The credential layer validates HTTPS URLs, header names, secret sizes, TLS material, custom auth JSON, and credential type; secrets are encrypted or masked for public reads. Provider settings support model defaults, base URL, host allowlists, private-network flags, timeouts, output limits, and monthly caps.	This is a strong start for secret governance. The production packet should still show the envelope key management plan, secret rotation process, and proof that plaintext keys are not shipped in source, logs, or reviewer bundles.
Webhook execution	<code>/api/workflows/webhook/{panelId}/{workflowId}</code> accepts POST only, rejects query payloads, requires content length, enforces body-size limits, verifies HMAC signature and timestamp, derives or honors idempotency keys, rate-limits authenticated executions, requires active workflow status, loads the live published version, persists execution/audit records, and stores idempotency results.	This is the right shape for external automation intake. Reviewers should see HMAC failure, replay, duplicate, inactive-workflow, and rate-limit negative tests before production use.
Defaults and memory	Workflow defaults cover execution timeout, webhook rate limits/idempotency TTL, and AI memory retention. Memory browse/clear is permissioned separately.	The model recognizes that AI memory and webhook behavior are operational policy settings, not hidden implementation details.

The professional claim for the workflow plane is therefore: workflow automation is architecturally present and several sensitive operations are already server-authorized and storage-driver-backed, but the UI read path and chat/workflow realtime surfaces must be completed for a fully self-hosted administrative product. In review materials, this should be framed as a governed automation plane with explicit secrets, provider, versioning, execution, webhook, and audit controls.

14.11 Workflow settings and policy console

The workflow settings console deserves separate treatment because it is where workflow automation becomes governable. The reviewed route first verifies a server session, re-

solves the user through the storage driver, checks panel access, computes workflow role, and passes only the allowed actions to the client. The tab set then exposes Permissions, Defaults, Models, Credentials, and Versioning. These are not cosmetic settings; they define who may build, run, publish, connect tools, retain AI memory, and manage secrets.

Workflow settings tab	Reviewed control	Review expectation
Permissions	Admin users can list panel workflow users and assign explicit workflow roles. The API blocks self-demotion, verifies the target belongs to the same panel or is globally manageable, prevents inappropriate platform-admin role management, and writes <code>Workflow_Role_Updated</code> audit logs.	Acceptance tests should prove viewer/editor/operator/admin boundaries, self-demotion rejection, cross-panel rejection, and audit visibility for role changes.
Defaults	Admin users can update execution timeout, retry defaults, webhook rate limit, webhook idempotency TTL, AI max agent iterations, and AI memory retention. Values are validated against hardcoded min/max limits and updates write audit logs.	Pilot packets should include the authority's execution timeout, retry, webhook-rate, idempotency, and AI-memory retention policy.
Models	Admin users can view, save, enable, test, and delete provider configurations for the workflow runtime. The route enforces <code>model:view</code> and <code>model:update</code> , validates known provider IDs, encrypts keys, and records provider update/delete audit events.	Provider tests, failed-key behavior, model selection, monthly caps, and private-network base URL approval should be documented before live AI workflow use.
Credentials	Editors/operators/admins may view credentials according to role; only admins create/update/delete. Credential types include MCP, HTTP, AI, email, SMS, chat, weather, GIS, and agency portal. The credential layer validates HTTPS URLs, auth modes, custom headers, bearer/query/basic/custom secrets, and TLS material, then masks public reads.	The deployment packet should include a credential inventory by reference ID and type, not secret values. Rotation, revocation, and MCP/HTTP endpoint approval should be explicit.
Versioning	Versioning tab reads recent publish/restore events from audit logs. Publish and restore routes snapshot workflow drafts, load published versions for live execution, and preserve execution-source metadata.	Operators should treat workflow publication like a change-control event. Reviewer evidence should show draft-to-publish, restore, execution version, and audit trail.

The workflow builder's node palette is enumerated in full below. The catalog is generated directly from the reviewed source (the builder's template array and the connector registry, evaluated rather than transcribed) by `tools/gen_workflow_node_catalog.py`, so a future edition refreshes it in one regeneration step and the counts and descriptions cannot silently drift from the shipped product.

The reviewed builder palette declares **264 node templates**: 117 core nodes across 7 families (1 retained only as a hidden legacy alias so existing workflows keep resolving), plus 147 marketplace connector action nodes generated from the connector registry. Descriptions below are the in-product node descriptions from the reviewed source, not editorial paraphrase.

Node template	Palette name and in-product description
Trigger nodes (27)	
crisis-signal-trigger	Crisis Signal Trigger. Starts when a verified SOS, sensor, or incident stream event arrives.
webhook-intake	Webhook Intake. Starts from an external agency, municipal, SMS, or partner webhook.
manual-run	Manual Run. Starts a workflow manually for testing, drills, or operator-triggered actions.
schedule-trigger	Schedule Trigger. Runs checks on a fixed interval, like preparedness scans or daily reports. A cron expression (e.g. 0 9 * * 1-5) overrides the interval.
chat-trigger	Chat Trigger. Starts when an operator, responder, or citizen sends a chat message.
mcp-server-trigger	MCP Server Trigger. Exposes this workflow as an MCP tool so external AI agents (Claude, ChatGPT...) can call it. Activate the workflow, then POST JSON-RPC to <code>/api/workflows/mcp/{panelId}</code> .
rss-feed-trigger	RSS Feed Trigger. Polls any RSS/Atom feed on an interval and starts the workflow for new items.
sse-trigger	SSE Trigger. Subscribes to a Server-Sent Events stream and fires per event. Needs an on-prem / persistent runtime (serverless has no long-lived connections).
local-file-trigger	Local File Trigger. Watches a filesystem path for changes (added/changed/removed). On-prem only — requires a persistent host with disk access.
email-inbox-trigger	Email Inbox Trigger. Starts from a monitored inbox for partner reports and forwarded alerts.
sensor-threshold-trigger	Sensor Threshold. Starts when IoT, seismic, weather, or telemetry streams cross a threshold.
form-submission-trigger	Form Submission. Starts when a public intake form is submitted — citizen incident reports, damage assessments, or volunteer sign-ups.
cap-alert-trigger	CAP Alert. Starts on an inbound Common Alerting Protocol (CAP) message — IPAWS/EAS, MeteoAlarm, NWS, or any CAP 1.2 authority.
gdacs-alert-trigger	GDACS Multi-Hazard Alert. Polls the Global Disaster Alert & Coordination System (UN OCHA / EC JRC) for earthquakes, cyclones, floods, volcanoes and droughts.
mqtt-sensor-trigger	MQTT Sensor. Subscribes to an MQTT broker topic for IoT telemetry — flood gauges, gas/CBRN sensors, structural-health or weather stations.

Node template	Palette name and in-product description
message-queue-trigger	Message Queue. Consumes events from a message bus — Kafka, AMQP/RabbitMQ, AWS SQS, NATS or Redis Streams — for high-volume incident fan-in.
emergency-call-trigger	Emergency Call. Starts on an inbound emergency call from a PSAP/CAD or telephony provider (112/911/119) — optionally transcribing the audio for triage.
geofence-trigger	Geofence Event. Starts when a tracked responder, asset or device enters, exits or dwells in a monitored zone — evacuation breaches, arrivals, departures.
db-change-trigger	Database Change. Polls a database table/collection for new or changed rows (change data capture) — new incidents, casualty updates, resource stock moves.
seismic-event-feed	Seismic Event Feed. Polls a seismic event feed from a configurable source (or your own endpoint). Source-agnostic — pick a provider or set a custom Feed URL.
kandilli-seismic-feed	Kandilli Bulletin (legacy). Legacy node — use Seismic Event Feed with the Kandilli source instead. <i>(hidden legacy alias)</i>
weather-warning-feed	Weather Warning Feed. Pulls a severe-weather warning feed from a configurable source (or your own endpoint).
wildfire-feed	Wildfire Feed. Pulls a wildfire/forest-fire status feed from a configurable source (or your own endpoint).
road-closure-feed	Road Closure Feed. Pulls a road-closure feed from a configurable source (or your own endpoint).
multi-hazard-feed	Multi-Hazard Alert Feed. Polls a multi-hazard aggregator (GDACS) that reports earthquakes, cyclones, floods, wildfires, volcanoes and droughts in one feed, each with a colour alert level. Optionally filter by hazard type / alert level.
social-signal-feed	Social Signal Feed. Pulls public distress/incident signals from a social or news source (RSS/Atom keyless, or Bluesky). Filters by keywords and writes normalized signals — an independent early-detection channel and a correlation corroborator.
flood-monitor	Flood / Hydro Monitor. Polls a hydrological source for river discharge / gauge levels. Open-Meteo (GloFAS) gives a genuinely predictive forecast crest for a point; USGS gives real-time US gauge observations by station id. Threshold the summary downstream for early flood warning.
AI nodes (18)	
sentinel-classifier	Sentinel Classifier. Classifies urgency, extracts location, and produces routing hints.
ai-agent	AI Agent. Plans next steps with selected memory, tools, and response constraints. Loops with function calling.
ai-chat-model	Chat Model. Supplies the reasoning model for an AI Agent. Connect it to the Agent's Chat Model slot.
simple-memory	Simple Memory. Adds conversation memory to an AI Agent using workflow-scoped storage.
crisis-toolkit	Crisis Tool Kit. Exposes built-in crisis tools to an AI Agent: weather, protocol lookup, location resolution, and severity scoring.
mcp-tool-server	MCP Tool Server. Discovers external MCP tools and exposes them to a connected AI Agent.

Node template	Palette name and in-product description
crisis-orchestrator	Crisis Commander (Multi-Agent). Coordinates specialist sub-agents (triage, situation, knowledge, dispatch) into one synthesised, RAG-grounded incident brief.
rag-retrieval	RAG Retrieval. Pulls relevant SOPs, maps, shelter data, or previous incident context.
incident-summarizer	Incident Summarizer. Condenses noisy reports into a short operational brief.
ai-transform	AI Transform. Transform the input with a natural-language prompt (a mini LLM chain).
sentiment-analysis	Sentiment Analysis. Classify text as positive / negative / neutral with a confidence score (LLM, lexical fallback).
ai-guardrails	AI Guardrails. Rule-based screen for LLM input/output: prompt injection, PII, blocked phrases, off-allowlist URLs, max length. Routes true (passed) / false (violations, listed on payload.guardrails).
structured-output-parser	Structured Output Parser. Extract JSON from (LLM) text and validate it against a schema. Sets parse_error when it doesn't match.
model-selector	Model Selector. Pick a model based on a condition (e.g. use a strong model only for critical incidents). Sets 'model' for downstream AI nodes.
reranker	Reranker. Reorder an items array by relevance to a query (lexical keyword overlap). Keeps the top-K.
tool-router	Tool Router. Chooses a downstream tool path based on intent, risk, and confidence.
image-damage-analyzer	Image Damage Analyzer. Uses multimodal AI to detect structural damage, fire levels, or road blockage from citizen photos.
llm-translation	AI Translation. Translates SOS messages, partner briefs, or SMS alerts between Turkish and English.
Flow-control nodes (21)	
sub-workflow-trigger	When Called by Workflow. Entry point for a reusable sub-workflow — receives the payload passed by a parent workflow's Sub-Workflow Call.
severity-gate	Severity Gate. Branches or blocks low-confidence events before dispatch.
if-gate	IF Gate. Splits the workflow into true and false paths using one expression.
switch-router	Switch Router. Routes incidents by region, severity, source, or capability match.
filter-items	Filter Items. Keeps only matching responders, shelters, reports, or sensor events.
wait-delay	Wait / Delay. Pauses execution for retries, cooling periods, or timed escalations.
merge-streams	Merge Streams. Combines parallel reports, AI outputs, and responder availability data.
loop-batches	Loop Batches. Processes responders, neighborhoods, or notifications in controlled batches.
weather-checker	Weather Condition Checker. Checks real-time weather and wind limits for coordinates before deploying rescue units.
retry-with-backoff	Retry with Backoff. Emit an exponential backoff schedule downstream HTTP / SMS nodes can consult.

Node template	Palette name and in-product description
rate-limit	Rate Limit. Token-bucket throttle — blocks the step when calls/minute exceed the cap.
idempotency-key	Idempotency Key. Block re-runs for the same key (e.g. incident.id) within a TTL window.
stop-and-error	Stop and Error. Explicitly halt the workflow with a structured reason — useful as the fail branch of a gate.
respond-to-webhook	Respond to Webhook. Return a custom HTTP status, headers, and body to the webhook caller. Requires the trigger's Response Mode = 'Respond to Webhook node'.
geofence-check	Geofence Check. Point-in-polygon test for evacuation zones, hazard areas, or jurisdictional boundaries.
error-trigger	Error Trigger. Activates only when the payload contains an error — use as the entry of a recovery branch.
sub-workflow-call	Sub-Workflow Call. Invoke another workflow by id and forward the current payload as input.
wait-for-webhook	Wait for Webhook. Emit a callback URL and pause until an external system pings it back (or timeout).
try-catch	Try / Catch. Branch into 'true' if a prior step left an error on the payload, otherwise 'false'.
polygon-contains	Polygon Contains. Tests a point against a LIST of polygons — first match wins (use for jurisdiction lookup).
jwt-verify	JWT Verify. Verify a JWT against a shared secret — branches 'true' on valid, 'false' on invalid.
Data nodes (26)	
set-fields	Edit Fields. Sets workflow data using manual mappings or a JSON output object.
code-step	Code Step. Runs a controlled script-like transform against workflow data.
json-parse	JSON Parse. Parses raw webhook, email, or partner payload text into structured data.
deduplicate	Deduplicate. Drops duplicate items using all fields, selected fields, or exclusions.
geocode-location	Geocode Location. Turns text location, coordinates, or landmarks into normalized map data.
split-out	Split Out. Emit each element of an array as its own item — n8n-style split-out.
aggregate-items	Aggregate Items. Roll an array of items back into a single field — inverse of Split Out.
summarize-items	Summarize Items. Summarizes item arrays with count, sum, average, min, max, unique count, append, or concatenate operations.
template-render	Template Render. Render a Mustache-style template with payload values into a target field.
hash-sign	Hash / Sign. Compute SHA-256, SHA-1, MD5, or HMAC-SHA-256 of a payload field.
csv-parse	CSV Parse. Parse a CSV string into an array of objects (first row optionally treated as header).
crypto-random-id	Random ID. Generate a UUID v4 (or a short hex id) for correlation, idempotency, or filenames.
timestamp-now	Timestamp Now. Write the current time to a payload field in ISO, Unix, or locale format.

Node template	Palette name and in-product description
data-store-query	Data Store Query. Read the panel's own stored data (incidents, records, audit logs...) into the flow. Read-only, filterable; works on cloud (Firestore) and on-prem (Postgres).
data-table	Data Table. Read/write rows in a panel-scoped named table — dedupe ledgers, counters, state shared between runs and workflows. Persistent; works on cloud (Firestore) and on-prem (Postgres).
distance-haversine	Geo Distance. Great-circle distance between two coordinates (km). Used for dispatch radius checks.
sort-items	Sort Items. Sort an array using n8n-style simple, random, or sandboxed code sorting.
limit-items	Limit Items. Keep only the first or last N items of an array — caps fan-out for broadcasts.
compare-datasets	Compare Datasets. Compare two item streams with n8n-style match fields, difference handling, and output groups.
date-time	Date & Time. Parse / format / add / subtract / diff dates with TZ-aware math.
xml-parse	XML Parse. Parse an XML or RSS string into JSON (preserves attributes with @_ prefix).
markdown	Markdown. Convert between Markdown and HTML with n8n-style mode, destination, and parser options.
html	HTML. Generate HTML templates, extract content with CSS selectors, or convert items into an HTML table.
reverse-geocode	Reverse Geocode. Coordinates → human-readable address (default: OpenStreetMap Nominatim).
signed-url-issue	Signed URL. Emit an HMAC-signed time-limited URL — share files/reports without exposing storage.
pdf-render	PDF Render. Render a SITREP-style PDF from a title + body — output is base64 ready for email/storage.
Core integration nodes (9)	
http-request	HTTP Request. Calls any external API with method, URL, auth, and response handling.
team-chat-notify	Team Chat Notify. Posts an incident update to an operations chat or partner channel.
send-email-sms	Send Email / SMS. Sends a templated notification through email, SMS, or both.
sms-broadcaster	Emergency SMS Broadcaster. Broadcasts critical location-based warnings to all citizen cell numbers in the area.
agency-portal-integration	Agency Portal Integration. Sends incident data directly to an external agency or emergency portal gateway.
seismic-event-fetch	Fetch Seismic Events. Action-mode seismic fetch (same parser as the Seismic Event Feed) for use mid-graph — e.g. pull a second network (USGS) as a cross-source corroborator after a primary seismic trigger.
multi-hazard-fetch	Fetch Multi-Hazard Alerts. Action-mode multi-hazard fetch (same parser as the Multi-Hazard Alert Feed) for use mid-graph — e.g. pull GDACS as a corroborator after a primary trigger.
social-signal-fetch	Fetch Social Signals. Action-mode social/news fetch (same parser as the Social Signal Feed) for use mid-graph — pull social distress posts as a correlation corroborator after a primary trigger.

Node template	Palette name and in-product description
satellite-imagery-feed	Satellite Imagery. Fetches satellite/aerial imagery for an area of interest and feeds it to the Image Damage Analyzer. Multi-provider (Sentinel Hub, NASA GIBS, Maxar, Planet) plus a custom endpoint for agency (AFAD/FEMA) ArcGIS/WMS systems.
Response nodes (15)	
dispatch-field-teams	Dispatch Field Teams. Notifies selected field teams and writes the dispatch event.
create-incident-record	Create Incident Record. Creates a normalized incident and updates the dashboard timeline.
voice-call-escalation	Voice Call Escalation. Starts a call escalation path for critical incidents or no-response cases.
human-approval	Human Approval. Creates a review task before high-risk or irreversible workflow actions.
assign-responders	Assign Responders. Assigns capable responders based on role, distance, and current load.
write-audit-log	Write Audit Log. Writes an immutable audit entry for compliance and after-action review.
gis-hazard-overlay	GIS Hazard Overlay. Correlates incident coordinates with active fault lines, flood basins, or population density.
escalation-chain	Escalation Chain. Emit a structured primary → backup → lead escalation plan with per-step timeouts.
notify-ack	Notify & Ack. Send a message to a roster and durably wait for every acknowledgment — 'acked' fires when all confirm, 'escalate' fires with the non-responders when the window expires. Chain on 'escalate' for tiered escalation.
hazmat-lookup	Hazmat Lookup. Map a UN number to the ERG response guide + hazard summary (embedded dataset).
situation-report	Situation Report. Assemble an ICS-209-lite sitrep object from named payload fields.
disaster-taxonomy	Disaster Taxonomy. Normalize free-text incident type to EM-DAT-lite codes (earthquake, flood, fire, ...).
tsunami-risk-check	Tsunami Risk Check. Deterministic Mw + depth + offshore check — branches into 'true' for at-risk events.
signal-correlation	Signal Correlation. Fuses a primary signal set (e.g. seismic events) with corroborating sources (social, other hazards, satellite) in space + time and scores confidence. Branches 'true' when max confidence ≥ threshold — the earliness/false-alarm lever.
anomaly-detector	Anomaly Detector. Baseline-deviation detection: flags a fast-rising series (rate), a value spiking above its mean (spike z-score), or a surge of events in a window (count — foreshock swarm / social-post surge). Branches 'true' when the score crosses the threshold.
Canvas notes (1)	
sticky-note	Sticky Note. Annotates and groups workflow areas with markdown-friendly notes.

Marketplace connector nodes (147). Every connector node shares one reviewed execution shape rather than bespoke code paths: a required credential reference resolved from the panel's encrypted workflow credential store, typed operation maps where the tar-

get service defines multiple operations, request timeouts, and outbound HTTP routed through the shared SSRF-guarded fetch layer described earlier in this section. The generated catalog currently covers: Slack Message, Twilio SMS, Gmail Send, Drive Search, Outlook Mail, Calendar Event, Sheets Append Row, Discord Message, AFAD Events, ArcGIS Query, Telegram Message, Teams Message, Mattermost Message, Google Chat Message, PagerDuty Trigger, WhatsApp Message, SendGrid Email, Mailgun Email, Notion Page, Airtable Record, Jira Issue, Trello Card, USGS Earthquakes, GDACS Alerts, ReliefWeb Disasters, Zoom Team Chat Message, Webex Message, Zulip Message, Rocket.Chat Message, Pushover Notification, ntfy Publish, Gotify Message, Matrix Message, LINE Push Message, Pushbullet Push, Opsgenie Create Alert, Datadog Post Event, Rollbar Report Item, Statuspage Create Incident, Better Stack Create Incident, Healthchecks.io Ping, GitHub Create Issue, GitLab Create Issue, PagerTree Create Alert, Squadcast Trigger Incident, HubSpot Contact, Zendesk Ticket, Freshdesk Ticket, ClickUp Task, Asana Task, Monday Item, Linear Issue, Intercom Contact, Pipedrive Deal, Vonage Send SMS, MessageBird Send SMS, Plivo Send SMS, Postmark Send Email, Mailjet Send Email, OpenWeatherMap Current, WeatherAPI Current, NWS Active Alerts, NASA EONET Events, EMSC Recent Quakes, Supabase Insert Row, Upstash Redis Command, Firestore Create Document, Firebase RTDB Write, Neon Insert Row, Turso Execute SQL, Xata Insert Record, Cloudflare D1 Query, Elasticsearch Index Document, InfluxDB v2 Write, CouchDB Create Document, Tinybird Ingest Event, NASA FIRMS Area Fires, EFFIS Forest Fires, USGS Water Gauges, GeoNet NZ Quakes, GeoNet NZ Volcanoes, GEOFON Global Quakes, USGS Elevated Volcanoes, NHC Active Storms, SWPC K-index, OpenAQ Air Quality, WAQI Air Quality, ReliefWeb Reports, HDX Data Search, IFRC GO Appeals, OpenFEMA Declarations, WHO Outbreak News, Nominatim Geocode, ORS Isochrones, GeoNames Lookup, airplanes.live Point, USGS Quake Detail, Ushahidi Posts, KoboToolbox Submissions, MeteoAlarm Warnings, USGS Did You Feel It, RapidPro Broadcast, Signal Send, Telerivet Send SMS, Ona Form Data, MeteoAlarm EDR Feed, Traccar Positions, Traccar Devices, OwnTracks Last Locations, what3words Convert, ORS Directions, GraphHopper Route, Overpass Nearby Facilities, HOT TM Active Projects, CDC Socrata Query, Safecast Radiation, AISHub AIS Positions, MarineTraffic Vessel Positions, VesselFinder Vessels, Global Fishing Watch Events, NOAA CO-OPS Water Levels, NOAA NDBC Buoy Observations, IOC Sea Level Monitoring, EMODnet Physics WFS, Datalastic Vessels In Radius, NOAA PTWC Tsunami Messages, UK Carbon Intensity, EIA Electricity Data, ENTSO-E Grid Data, Electricity Maps Live, NREL PVWatts, WattTime Signal Index, Elexon BMRS Demand, NESO Datastore Search, CAISO OASIS Query, EirGrid Grid Data, OpenEI Utility Rates, AfterShip Create Tracking, EasyPost Create Tracker, Shippo Create Shipment, TrackingMore Create Tracking, 17TRACK Register, ShipEngine Track, Onfleet Create Task, Routific Optimize, Samsara Vehicle Locations, Sendcloud Tracking.

The breadth is useful, but it raises the governance bar: live workflows should be allowed only after credentials, webhook secrets, provider settings, human-approval nodes, dry-run output, and publication version are reviewed.

14.12 Firestore rules and cloud-mode authorization

In cloud mode, Firestore rules implement panel access, rescue write constraints, and tamper-resistant boundaries:

- Dashboard users may read agency panels only when their role and agency binding, hierarchy relationship, or platform-admin flag permits access.
- Field-team/admin rescue writers may update their own agency panel only when their verified profile and custom role claim match the target `agencySlug`.
- Responder documents require matching `uid`, `agency`, and `responderDeviceId` constraints for non-platform writes.
- Signal documents must use a bounded ID format (`CC-` plus 24 lowercase hex characters), status `active`, source `ble`, schema version at least 1, and last-seen timestamps not earlier than first-seen timestamps.
- Audit logs, workflow credentials, webhook idempotency / rate-limit state, provider credentials, and model artifacts are not client-writable; server endpoints or admin SDK paths own those writes.
- Firebase Storage is fail-closed except for per-user avatar writes gated by authenticated self-ownership, assigned role, organization affiliation, image MIME type, and a 2 MB size cap. Crisis Sentinel model artifacts are denied at Storage Rules and served only through server-controlled signed URLs.
- The July 2026 rule set adds server-only key collections (mesh, messaging, hierarchy-channel, and prekey material), membership-gated hierarchy channels with immutable ciphertext logs, get-only call-room documents with listing denied and TTL expiry, device-ownership records with strict field allowlists, and owner-only presence settings; the rules are exercised by an automated emulator test script, which reviewer packets should cite as rules-regression evidence.

These rules are relevant to cloud deployments only. In on-prem deployments, equivalent authorization must be enforced by server routes, Keycloak/enterprise identity, Postgres access controls, and infrastructure policy.

14.13 Docker Compose evaluation stack

The Docker path is suitable for evaluation, lab pilots, and small single-node deployments. The `Dockerfile` uses a multi-stage Node 20 Alpine build: dependency stage, builder stage with `NEXT_OUTPUT_STANDALONE=true`, and runner stage that copies the standalone Next.js server, public assets, static files, Drizzle migrations, scripts, and runtime modules. The final container runs as non-root user `nextjs` with `uid/gid 1001` and exposes port 3000. Its container healthcheck calls `/api/health`; that endpoint is

intentionally shallow and reports server liveness plus version metadata rather than deep DB/Keycloak health.

Compose service	Image / role	Operational details	Data
postgres	postgres: 16-alpine	App database for on-prem storage driver and Keycloak database host in the evaluation stack. Uses <code>pg_isready</code> healthcheck.	Named volume.
keycloak	quay.io/keycloak/ keycloak:25.0	OIDC realm/client/user source for self-hosted auth. Starts with HTTP enabled and x-forwarded proxy headers.	Postgres-backed.
minio	minio/minio: latest	S3-compatible object storage for self-hosted artefact/file paths. Exposes API and console ports in evaluation mode.	Named volume.
redis	redis:7-alpine	Cache/queue support with password and append-only persistence.	Named volume.
app	Built Crisis Connect Web image	Runs Next.js standalone server in on-prem mode with Postgres, Keycloak, MinIO, Redis, license, update, and public URL environment.	Stateless.
health-reporter	Same app image, optional profile	Runs the installation health reporter loop and sends outbound metadata heartbeat if configured.	Stateless.
workflow-scheduler	Same app image, optional profile	Runs the workflow scheduler tick loop against <code>/api/workflows/scheduler/tick</code> ; the tick endpoint fails closed unless <code>WORKFLOW_SCHEDULER_TOKEN</code> is configured.	Stateless.

The one-machine evaluation flow is: create `.env`, start Postgres/Keycloak/MinIO/Redis, run migrations, bootstrap Keycloak, copy generated secrets into environment, then start the app. Production systems MUST replace sample passwords, avoid plaintext secret storage, and move secrets to a managed secret store.

14.14 Helm and Kubernetes production path

The Helm chart is the production-oriented self-hosted path. It defaults to two app replicas, ClusterIP service on port 3000, requests of 250 mCPU / 512 MiB, and limits of 1500 mCPU / 1 GiB. The chart injects non-secret configuration through a ConfigMap and sensitive values through a Kubernetes Secret. Config/secret checksums force pod restart when operational configuration changes.

The chart's migration Job runs as a Helm `pre-install`, `pre-upgrade` hook and executes `scripts/migrate.mjs`; this places the target schema before new app pods receive traffic. Optional CronJobs support license renewal (twelve-hourly), installation health reporting (five-minutely), and per-minute workflow-scheduler ticks. The chart intentionally does not ship Keycloak as a first-class production component; production deployments should use a Keycloak Operator, a dedicated Keycloak chart, or an external

compliant OIDC provider. Postgres, Redis, MinIO/S3, and Keycloak should normally be operator-managed for national-agency deployments.

14.15 Self-hosted operating responsibility model

The reviewed artefacts support self-hosting, but production readiness depends on how responsibilities are assigned. A professional deployment packet should distinguish application responsibilities from authority-operated infrastructure responsibilities.

Control area	Application / vendor responsibility	Authority / operator responsibility
Release integrity	Provide source identifier, image tag, build SHA, migration set, release notes, and signed license / update public-key configuration.	Verify image provenance, pin values files, approve migrations, and retain release manifests in the authority change record.
Identity and access	Provide Fire-base / Keycloak / enterprise OIDC integration points, server-session handling, role mapping, and dashboard access checks.	Own IdP policy, MFA / session policy, role assignments, responder / authority onboarding, and periodic access reviews.
Secrets and keys	Keep license / update private signing keys outside the app runtime; provide public verification key configuration and secret reference patterns.	Store app secrets, OIDC client secrets, database credentials, license tokens, and update configuration in managed secret systems.
Data platform	Provide storage-driver abstraction, Drizzle migrations, Postgres path mapping, and cloud-mode Firestore rules where applicable.	Operate Postgres backups / PITR, Redis, object storage, database monitoring, retention policy, and disaster-recovery drills.
Network and observability	Provide health endpoints, optional heartbeat reporter, and deployment hooks.	Enforce ingress / TLS / network policy, deep service monitoring, alerting, log retention, and incident response runbooks.

Control area	Application / vendor responsibility	Authority / operator responsibility
AI and workflow automation	Provide Crisis Sentinel provider validation, RAG ingestion/retrieval hooks, model-release metadata validation, workflow RBAC, credential envelope handling, AI generation/execution routes, webhook HMAC/idempotency controls, and audit hooks.	Approve AI providers, RAG knowledge sources, MCP endpoints, model/data residency, prompt and memory retention, model artefact provenance, credential rotation, automation/tool allowlists, and live-execution change policy.
Map and external services	Provide Leaflet-based map integration and document current public tile dependency.	Approve public basemap use or replace it with agency-hosted tiles for regulated or isolated deployments.

14.16 Licensing, updates, and health reporting

On-prem deployments can use signed annual, pilot, trial, or internal-test licenses. The license token is an offline-verifiable ES256 JWT checked with a public key deployed to the customer environment. The private signing key is never deployed to the app, Docker Compose, Helm, or git. License enforcement supports **observe** mode for pilots and **strict** mode when feature gates should block missing, expired, invalid, or unlicensed modules. The dashboard feature gate is enforced before `/api/dashboard/overview` returns data.

Self-hosted update management is also outbound-only. The app can fetch a signed JWS update manifest, verify it with an ES256 public key, and show authorized admins the latest version, severity, migration requirement, and Docker/Helm commands. The app does not execute remote update commands and does not store a manifest private key.

Installation health reporting is optional. A self-hosted deployment may send periodic metadata-only heartbeat reports to the central platform. The receiving endpoint is available only in cloud mode and only when a shared bearer secret is configured. Payloads are bounded to installation metadata such as installation ID, customer/panel labels, mode, version, image tag, build SHA, region, health timestamp, license status, update status, and notes. User tokens, OIDC client secrets, license private keys, provider credentials, and emergency message content are not part of the health report.

14.17 Enterprise SSO and identity governance

The web line supports multi-tenant enterprise OIDC. Each agency can register a `SSOC` `connections/{providerId}` document or a corresponding Postgres row with issuer, metadata URL, client ID, allowed email domains, role mappings, JIT provisioning policy, and email-verification requirement. Client secrets should be referenced through environment variables or encrypted with `SSO_SECRET_ENCRYPTION_KEY`; plaintext provider secrets **MUST NOT** be stored in app source or dashboard documents.

The Keycloak bootstrap script creates or updates a local evaluation realm, adds `admin`, `authority`, and `fieldTeam` roles, configures a web OIDC client with PKCE, configures a service-account admin client, seeds an enterprise SSO provider record, and creates a test admin user. This is an evaluation/bootstrap convenience, not a substitute for institutional IAM change control.

14.18 Production hardening requirements for the web control plane

Before a production authority deployment, the web control plane **SHOULD** satisfy the following hardening checklist:

- Build with on-prem public deployment mode and run with on-prem server deployment mode when Firebase-free infrastructure is required.
- Store all production secrets in sealed/external secret systems or equivalent managed secret stores. Never publish `.env`, `.env.local`, `.secrets`, license signing keys, update signing keys, or OIDC client secrets.
- Use operator-managed Postgres with backups and point-in-time recovery, operator-managed Redis, operator-managed MinIO/S3 with object-lock or equivalent audit retention where needed, and dedicated Keycloak/OIDC infrastructure.
- Replace public basemap dependencies with agency-approved tile services for isolated or regulated deployments.
- Treat `/api/health` as liveness/readiness only; add deep monitoring for Postgres connectivity, OIDC discovery/token exchange, Redis, object storage, license status, and update-manifest reachability.
- Keep `LICENSE_ENFORCEMENT_MODE=observe` during pilots unless the authority explicitly accepts feature blocking behavior.
- Keep unsigned update manifests disabled outside local testing.
- Validate that all dashboard, settings, AI/chat, and workflow screens required by the authority are routed through storage-backed server APIs before representing the admin experience as fully self-hosted.
- Treat AI provider keys, RAG knowledge sources, MCP credentials, workflow webhook secrets, model-release metadata, mobile model artifacts, and prompt/memory records as regulated operational secrets or sensitive records with explicit retention, rotation, and audit policy.

14.19 Production acceptance criteria for dashboard deployment claims

The following gates should be met before external materials describe the web dashboard as a production-ready self-hosted control plane for an authority deployment:

Gate	Minimum evidence	Why it matters
Mode alignment	Build-time and run-time deployment mode are both documented; test run confirms on-prem dashboard loads through server APIs.	Prevents a mixed cloud / on-prem configuration from being misrepresented as Firebase-free.
Identity proof	Keycloak or enterprise OIDC login, role mapping, dashboard access denial, and panel-scope tests are recorded.	Confirms authority IAM controls the dashboard rather than client-side state.
Data-plane proof	Postgres migrations applied, panel/responders/signals loaded through storage driver, and backup/restore procedure documented.	Confirms dashboard state is self-hosted and recoverable.
Secret handling proof	Sanitized environment template, secret inventory, and managed-secret destination are included; private signing keys are absent from runtime.	Prevents accidental publication of operational credentials or signing material.
Settings/ AI/ workflow proof	Required Settings tabs, Crisis Sentinel provider/ RAG/ model-release controls, AI assistant persistence, workflow lists, workflow settings tabs (Permissions, Defaults, Models, Credentials, Versioning), node catalog, dry-run/ live execution, webhooks, credentials, provider config, and memory controls are tested through server/ storage-backed paths or explicitly excluded from the on-prem claim.	Prevents hidden Firebase/ client dependencies from being discovered during authority acceptance testing.
Lifecycle proof	License verification, signed update manifest verification, migration command visibility, and rollback/ upgrade runbook are included.	Gives operators a controlled release and renewal process.

Gate	Minimum evidence	Why it matters
Observability proof	Liveness/readiness, deep dependency monitoring, audit log review, and optional health heartbeat policy are documented.	Makes operational failure modes visible before field use.

14.20 Current web open gaps

The reviewed web baseline is materially more than a public landing page: it contains an agency dashboard, protected routes, server-session APIs, self-hosted storage/auth drivers, license/update/health machinery, audit controls, workflow tables, an E2E messaging/call plane, external assistant channels, and Docker/Helm deployment paths. The main dashboard and server APIs are on-prem aware. The remaining risk is not whether a self-hosted dashboard path exists; it is whether every administrative surface required by an authority pilot has been moved onto the same storage-backed, self-hosted control model.

Since the June 6, 2026 edition, several previously listed gaps have verifiably closed: end-to-end test infrastructure now exists (Playwright workflow suites, including a Keycloak/SSO variant), rate limiting covers all new public surfaces, scheduler reliability has transactional multi-replica safety with retry/dead-letter handling, secrets-at-rest and runtime secret resolution moved to encrypted envelopes and a managed secret store, cross-org isolation was hardened in the security rules, ops CronJobs exist for licensing and health reporting, and audit coverage extends to assistant-channel configuration, MCP tool calls, and scheduled executions. The list below is the remaining open register.

- **Backup and restore.** No backup/restore scripts or tested restore procedure exist in the repository; the only backup guidance is a one-line operator note in the deployment guide (pg_basebackup to S3 plus PITR). For an authority deployment this is a hard production gap: operator-managed Postgres backup/PITR is an operator responsibility, but the vendor packet should ship a tested restore procedure.
- **Independent assurance.** The public trust page itself discloses that no independent security audit or penetration test has been performed (top roadmap item), and that ISO 27001 / SOC 2 certification and a paid bug bounty do not exist. Institutional packets MUST NOT imply otherwise.
- **KVKK disclosure.** The Turkish data-protection disclosure page is drafted but feature-flagged off pending legal review; it cannot be cited as published.
- **Release versioning discipline.** No git tags or changelog; the package version is static while the codebase moved by roughly a hundred thousand lines. Review claims must bind to image tag and build SHA.
- **E2E test breadth.** The new Playwright infrastructure is workflow-focused and narrow; dashboard, Settings, messaging, and assistant-channel journeys still need coverage.

- **Settings surface completeness.** User management, licensing, on-prem update status, certificates, audit writes, audit retention, and Crisis Sentinel provider/RAG/model-release configuration have meaningful server/API backing. Account/profile, notification, organization-role editing, scoped theme/config screens, and some security-log reads still use Firebase client listeners or writes and should be migrated before a full on-prem claim.
- **Crisis Sentinel governance packet.** Provider selection, built-in quota limits, BYO provider caps, RAG source approvals, knowledge-ingestion source IDs, multi-agent assessment policy, mobile model-release hashes, and model artefact storage paths should be documented before authority acceptance.
- **AI assistant persistence.** The /ai assistant is integrated with dashboard context and workflow generation, but chat sessions, message history, and related library/search behavior still rely on Firestore client paths. On-prem pilots should either exclude AI chat persistence from the claim or implement storage-backed server APIs.
- **Workflow UI read path.** Workflow execution, generation, credentials, providers, webhooks, defaults, memory, and settings authorization have substantial server-side controls. The main workflow list, run-history, and adjacent chat/listener surfaces still need storage-backed read/subscription paths for a fully Firebase-free workflow console.
- **Workflow settings policy proof.** Permissions, Defaults, Models, Credentials, and Versioning should be exported as an authority policy snapshot: role matrix, retry values, timeout values, webhook rate, idempotency window, provider enablement, credential reference inventory, publication trail, restore trail, and execution evidence.
- **AI governance evidence.** Authority packets should cover provider keys, self-hosted base URLs, private-network access, MCP endpoints, workflow credentials, webhooks, AI memory, RAG knowledge, agent outputs, model provenance, quotas, tool allowlists, and mobile release metadata, with retention, secret-management, and audit evidence.
- **Map isolation.** The current dashboard map uses public Carto basemap tiles. Regulated, disconnected, or national-agency deployments should define an agency-approved tile source or self-hosted tile service.
- **Production evidence packet.** The web line should ship with image provenance, build SHA, migration record, sanitized environment template, secret inventory, SSO provider manifest, Settings/AI/workflow route coverage, Helm/Compose values, and rollback/upgrade runbook before production review.
- **Correct current claim.** The defensible claim today is *main dashboard and selected administrative/workflow APIs support on-prem mode, not the entire administrative product is fully Firebase-free.*

15 Threat Model

15.1 Assets to protect

- Message confidentiality and integrity on every transport, including the internet relay path.

- Long-term identity keys, Signal session state, and prekey material on device and in the backend directory.
- Victim identity and location privacy (especially in SOS mode).
- Responder authenticity (preventing malicious impersonation).
- Sender authenticity on the relay path (the backend must not be able to forge messages between contacts).

15.2 Adversary capabilities

- Passive RF eavesdropping on BLE/Classic and Wi-Fi Aware local channels.
- Active attacks: replay, spoofing, connection jamming, device flooding.
- A compromised or coerced relay/push/TURN infrastructure operator: assumed able to read, delay, drop, reorder, or attempt to forge relayed documents and to observe routing metadata (identifiers, timing, sizes, push tokens). Confidentiality and sender authenticity are enforced end-to-end against this adversary; metadata exposure is a residual risk handled by policy.
- Directory-abuse attempts: phone-number harvesting, directory poisoning, and enumeration of anonymous identities (countered by opt-in verified-claim indexing, matches-only lookup, SPAKE2's one-guess-per-handshake property, and non-discoverable anonymous keys).
- Downgrade pressure: an attacker or misbehaving server attempting to force v3 sessions back to weaker envelopes (countered by the anti-downgrade pin and fail-closed receive paths).
- Control-plane disruption attempts against optional cloud synchronization endpoints.
- Physical presence in the affected area.

15.3 Out-of-scope (explicit)

- Full device compromise (malware with OS-level privileges).
- Coercion attacks on users.
- Nationwide-scale RF jamming.
- Global passive network observation and large-scale traffic-analysis correlation across the internet path.

16 Privacy Considerations

Privacy is a first-class concern in emergency systems because RF traffic can leak presence and patterns even when payloads are encrypted.

16.1 Metadata minimization

- Minimize stable identifiers in advertisements; prefer rotating, unlinkable pseudonyms where feasible.
- Avoid sending precise location unless explicitly enabled; support coarse location or “location withheld” states.

16.2 Internet-path presence and registration inventory (hybrid baseline)

The hybrid line introduces cloud-visible behaviors that institutional privacy review **MUST** inventory explicitly, because several are active by default:

- **Default-on registration.** On every app start with connectivity, the client performs an anonymous backend sign-in, publishes its identity public key (non-discoverably), registers a push token, and (on current builds) publishes Signal prekeys. No content or contact graph is uploaded by this step, but it creates a stable pseudonymous presence signal toward the backend.
- **Opt-in discoverability.** Phone / username directory listing requires explicit opt-in with verified-claim indexing; address-book discovery is matches-only with no server-side query storage; QR pairing remains the documented most-private path.
- **Presence signals.** Last-seen sharing is account-level opt-in stored owner-only; typing indicators and read receipts operate within E2E channels; SOS cloud uplink and contact escalation transmit operational location by design and only during an active SOS.
- **Analytics and crash reporting.** A product-analytics event layer exists (screen views and feature events); the implementation records transport / method / result / duration dimensions without user identifiers. Pilot packets **SHOULD** verify this claim with an event-schema export and treat analytics policy as a deployment decision. Build-level references on the current line additionally indicate a crash-reporting integration (Crashlytics); pilot packets **MUST** confirm whether crash reporting is active in release builds and, if so, inventory it as a separate telemetry channel, including what device and state data leaves the device on a crash.
- **Profile avatars.** Optional profile photos are stored as ordinary objects in project storage buckets and referenced by URL (clients pin avatar URLs to project buckets); they

are server-visible profile data rather than end-to-end encrypted content and should be inventoried as such.

- **Verification SMS provider.** Opt-in directory verification codes are delivered as SMS one-time passwords through a third-party provider invoked server-side (currently Twilio); opted-in phone numbers therefore transit that provider, which **MUST** appear in the deployment's data-processor inventory alongside the push and TURN providers.
- **On-device diagnostics.** Both mobile platforms maintain a file-backed connectivity/messaging diagnostics log (Android `conn_diag`; iOS parity implementation); pilot policy should govern its retention and anonymization, and pilot packets **MAY** use it as a supporting instrumentation artifact.
- **External feeds.** The Recent Disasters surface fetches public feeds (GDACS, AFAD, Kandilli, USGS, EMSC) when online, with region resolution falling back through GPS, IP, SIM, and locale signals; the surface is read-only and sends no user content to feed providers, but feed requests are ordinary internet traffic visible to those providers.
- **Metadata residue.** Even with ciphertext-only relay, the backend observes routing identifiers, timing, envelope sizes, and push tokens; TURN relays observe call endpoints' network addresses. Deployments with elevated threat models should weigh this residue explicitly and may prefer QR-only, offline-dominant operating profiles.

16.3 Rescue Mode trade-offs

SOS beaconing improves discoverability but can increase the risk of unwanted tracking. The design should define a user-visible consent model (e.g., explicit SOS toggle), include rate limiting, and document what information is emitted in each mode.

16.4 On-air metadata inventory (pilot policy matrix)

To make privacy controls auditable, pilot deployments should maintain an explicit inventory of over-the-air metadata by mode.

Operational note. Device name, service UUID pattern, and scan/advertise cadence can create radio fingerprints. Pilot reports should explicitly state advertisement cadence policy, jitter usage, and whether any stable identifiers were observed in packet captures.

16.5 Data retention

Device logs and message history should have clear retention defaults, a user-accessible deletion mechanism, and privacy-preserving identifiers for pilot evaluations.

Current Android baseline behavior:

- Message records use unique message UUID indexing to reduce duplicate persistence risk.

Mode	On-air fields	Rotation / timing policy	Primary privacy control
Onboarding window	Temporary discovery identity used for contact pairing workflow.	Strictly time-bounded (15 s primary + 8 s fallback), then disabled outside onboarding flow.	Pairing-only exposure window; no persistent broadcast after onboarding timeout.
Normal readiness	Service-level discovery signal only; no personal profile attributes in advertisement payload.	Address/randomization handled by OS BLE privacy mechanisms; deployment should avoid stable custom identifiers.	Minimize linkability by suppressing human-readable identifiers in routine advertisements.
SOS mode	Connectable Rescue service UUID (0xCC00) required for responder discoverability.	SOS broadcast is user-triggered and bounded by explicit mode state; interval jitter SHOULD be used to reduce fingerprinting.	Discoverability is intentionally elevated; user consent and clear UI state are required.
Connected secure exchange (RFCOMM contacts / Rescue secure chat / responder mesh)	Encrypted payload frames; metadata limited to transport-visible timing/length artifacts.	Session keys refreshed per connection/re-handshake boundary.	Confidential payloads with replay/duplicate guards; no plaintext identity claims on-air.
Public GATT general channel (opt-in)	Application packet fields include sender label, message body, IDs, hop/protocol metadata.	Enabled only by explicit operator setting; SHOULD be disabled when not needed.	Treat as non-confidential coordination channel; do not transmit sensitive personal/operational data.
SPAKE2 add-by-number beacon	Generic “open to add” service UUID only; no number-derived material on air.	Off by default; auto-enabled only after a phone number is added; never overrides explicit opt-out.	Harvest-resistant by design: a wrong number guess fails at key confirmation and reveals nothing.

- Call-event history is explicitly trimmed to a bounded window (200 events per session).
- Local sensitive keys use encrypted preference storage.
- Relay-side retention is bounded by design: relayed ciphertext envelopes are deleted on recipient acknowledgement, and unacknowledged envelopes are purged on a TTL by scheduled cleanup.

Deployment note. In the v1.1.7 baseline manifest, `allowBackup=false` is enforced. Institutional production builds should keep this posture (or stricter equivalent), explic-

itly govern any device-level backup exception policy, and verify that secure preferences/message databases remain outside unintended extraction paths.

16.6 Pilot log anonymization template

Log field class	Minimum anonymization rule
Device identifiers	Replace with run-scoped pseudonymous IDs; keep reversible mapping in separately protected access-controlled file.
User/contact labels	Remove plaintext names and handles; keep only role labels (victim/responder/tester).
Location records	Quantize to mission-approved precision (for example coarse grid or distance bin) unless fine precision is explicitly required for a signed test case.
Timestamps	Preserve timing deltas for latency analysis; if required, offset absolute clock values consistently per run.
Evidence artifacts	Attach checksum manifest and anonymization method declaration for each shared artifact bundle.

17 Evaluation Plan

This section defines an evaluation methodology appropriate for academic review and institutional pilots (e.g., AFAD-like deployments). Reported results should include device models, OS versions, and environmental conditions.

17.1 Test environments

- Open area / line-of-sight.
- Indoor (multi-room) with typical building materials.
- Urban canyon / mixed multipath.
- Rubble-like obstruction (simulated or controlled training site).

17.2 Scale and instrumentation

- Devices: 2–20 smartphones (mixed vendors), repeated trials.
- Logging: timestamps, RSSI, connection events, message IDs, battery level.
- Safety: privacy-preserving test identifiers; no real PII in logs.

17.3 Metrics

- Latency: time from send to receive (median, P95).
- Throughput: sustained payload rate over RFCOMM (when applicable).
- Success rate: fraction of messages delivered within a time bound.
- Range: maximum distance for stable delivery per scenario.
- Battery: drain rate in standby discovery vs. active session, including hybrid standby with internet registration and push listeners active.
- Authorized mesh readiness: activation success, peer-auth pass rate, and secure group-chat continuity for Wi-Fi Aware responder sessions.
- Internet-path delivery: relay delivery success and latency under degraded networks, push wake-up success rate across OEM power policies and iOS background states, transport-handover continuity (Bluetooth ↔ internet on one conversation), and acknowledgement-deletion verification.
- Internet calls: setup success, TURN-fallback rate, media continuity under network transitions, and PushKit/Telecom wake behavior.

17.4 Controlled standby power comparison (Pixel 7a, March 15, 2026)

A controlled paired measurement was recorded on March 15, 2026 on Google Pixel 7a (Android 13, app 0.9.9-debug) under airplane mode with Wi-Fi disabled, Bluetooth enabled, and the screen asleep. Because the device remained AC-powered during capture, the results should be interpreted as comparative device-power evidence rather than as a direct unplugged battery-runtime claim.

Two raw Perfetto traces were collected over matching windows of approximately 300 s each: (i) baseline idle with the app force-stopped, and (ii) app standby with Crisis Connect launched, backgrounded, and its RFCOMM foreground service left active while waiting for messages. Direct parsing of the raw traces yields 70,068,420 μ Ws total device energy in baseline idle (233.6 mW average device power) versus 74,287,841 μ Ws in app standby (247.7 mW average device power).

The resulting incremental standby overhead is therefore approximately 4,219,421 μ Ws over the 300 s window, or 14.1 mW average additional device power. The largest positive deltas appeared in Display (2.02 mW), CPU Mid (1.69 mW), Memory (1.69 mW), CPU Little (1.63 mW), CPU Big (1.13 mW), and a small WLAN increase (0.37 mW). Interpreted conservatively, this indicates a modest standby overhead rather than an always-on high-drain profile.

This evidence should not be relabeled as a pure SOS-beacon figure. The measured standby run left the RFCOMM foreground listener active and did not isolate the narrower BLE-advertising/GATT-server-only victim posture. If the document wants a numeric claim

specifically for SOS BLE standby, a separate matched trace should be captured with the GATT SOS server advertising and no responder/client scan or relay load.

This comparison is substantially stronger than the earlier single-window snapshot because it provides both device-level totals and an app-attributable standby delta under matched conditions. It still does not close the power-impact gate, however: the build is debug-only, the device was AC-powered, the sample covers one phone model, and there is no active transport or voice-session comparison in this evidence set. Pilot sign-off still requires repeat runs on release-equivalent builds across representative devices with unplugged conditions documented explicitly.

17.5 Adjacent-device voice baseline (March 15, 2026)

A supplemental clean-baseline voice run was recorded on March 15, 2026 between two Android phones running **0.9.9-debug**: Samsung SM-J260F (Android 8.1.0) and Samsung SM-G990E (Android 16). The devices were kept adjacent to each other while a bidirectional RFCOMM voice call was maintained for approximately 126 s before an intentional operator hangup.

RfcommService log dumps from both devices show answered-call teardown on both sides (125,712 ms and 126,599 ms respectively), zero observed playout-underrun counters, zero observed loss counters, and zero observed CRC-drop counters in the periodic playback-jitter logs. Adaptive profile switching between WB and NB occurred during the session, and transient out-of-order indications were observed, but the call remained continuous and ended by operator hangup rather than transport failure.

This run is useful supplemental evidence for clean-condition call continuity, but it does not close the voice gate: the devices were adjacent, the builds were debug, the packet still lacks reviewer-grade one-way latency median/P95 extraction from this run, and there are still no degraded-range or obstacle scenarios attached.

An additional short-range voice run was then recorded on March 15, 2026 with two walls between the same devices. Both sides again completed the call by operator hangup after approximately 212 s, with zero observed loss, playout-underrun, or CRC-drop counters in the periodic playback logs. Relative to the adjacent-device baseline, the wall-obstructed run showed materially higher sustained jitter-buffer depth on both devices (approximately 199–203 ms average, with repeated peaks at 240 ms), which is consistent with a more stressed but still continuous link.

This second run strengthens the continuity evidence under mild obstruction, but it still does not close the voice gate: one-way latency median/P95 was not extracted, the build remained debug, and the packet still lacks a scenario matrix across distance bins, heavier obstruction, and explicitly degraded RF conditions.

A third debug instrumentation run on March 15, 2026 validated the new per-call receiver-observed packet-age logging path in the normal `com.auralis.crisisconnect` package. The same two phones completed another answered call by operator hangup after approximately 172 s. On teardown, SM-G990E reported 8,361 packet-age sam-

ples with min/median/P95/max 1057/1085/1298/1612 ms (average 1123.4 ms), while SM-J260F reported 8,587 samples all clamped to 0 ms. The asymmetry is best interpreted as handset wall-clock skew rather than literal greater-than-one-second transport delay. If the SM-G990E series is normalized by its minimum observed age, the excess median/P95/max are approximately 28/241/555 ms.

This instrumentation run improves the packet because it proves that per-call packet-age summaries can now be harvested directly from production-style debug logs, but it still does not close the voice gate: absolute one-way latency remains confounded by unsynchronized device clocks, the build remained debug, and the packet still lacks range-binned or explicitly degraded-RF scenarios with synchronized timing controls.

A fourth March 15, 2026 instrumentation run then enabled monotonic clock-sync probes during the call so that packet-age could be expressed against a calibrated remote clock offset rather than against raw handset wall clocks. The same SM-J260F and SM-G990E pair completed another answered call by operator hangup after approximately 64–65 s. On teardown, SM-J260F reported 3,149 clock-adjusted packet-age samples with min/median/P95/max 3/18/99/299 ms (average 33.1 ms), while SM-G990E reported 3,175 samples with min/median/P95/max 1/16/120/356 ms (average 33.0 ms). Both sides recorded 13 usable clock-sync samples and zero discarded sync observations; the best observed sync RTTs were 17 ms and 27 ms respectively.

This fourth run materially improves the voice packet because it replaces the earlier wall-clock-skew caveat with a reproducible clock-adjusted one-way packet-age distribution on both handsets. It is still supplemental rather than gate-closing evidence, however: the build remained debug, the evidence covers one device pair and one indoor setup, and the packet still lacks a broader degraded-condition matrix and explicit packet-loss accounting tied to the latency run itself.

17.6 Range and RSSI reporting standard

To keep cross-pilot comparisons consistent, range evidence should be reported in fixed bins rather than only as single maximum values. This binning policy aligns with BLE RSSI measurement practice in localization literature.[8, 9]

Dimension	Reporting requirement
Distance bins	Report outcomes at minimum for 0–10 m, 10–20 m, 20–30 m, 30–50 m, 50–80 m, and 80–120 m where feasible.
RSSI bins	Report sample counts and delivery performance for at least five bins: > –60 dBm, –60 to –70 dBm, –70 to –80 dBm, –80 to –90 dBm, and < –90 dBm.
Per-bin statistics	Include attempts, success rate, median latency, and P95 latency per scenario and per bin.
Environmental annotation	For each run, include obstacle class, motion assumption, and known interference notes.

17.7 Evidence Matrix (Target vs. Pilot Evidence)

17.8 Acceptance Criteria (Pilot Gate)

Before pilot sign-off, deployments should define and approve explicit acceptance gates. Suggested initial thresholds are:

- **Message delivery success:** at least 95% delivery within 30 seconds in indoor/urban scenarios; at least 85% in rubble-like conditions.
- **Text latency:** end-to-end message latency P95 less than or equal to 5 seconds under planned operating range.
- **Voice quality envelope:** one-way latency median less than or equal to 150 ms and sustained packet loss less than or equal to 3% during active calls.
- **Power impact:** active-mode battery drain must stay within mission limits versus the standby discovery baseline.
- **Operational readiness:** responder authentication and SOS interaction must pass 100% of scripted validation scenarios.
- **Public-channel abuse containment:** malformed/out-of-window packet suppression must remain at or above 99% in stress runs, and unverified responder announcement payloads must never render as trusted alerts.
- **Internet-path readiness (when hybrid features are in pilot scope):** relay delivery at or above 99% within 10 seconds on reference networks with 100% queue recovery after reconnect, push wake-up at or above 95% across the approved device matrix, 100% conversation continuity across transport handovers, call setup success at or above 95% with TURN fallback engaging on all direct-path failures, and completed safety-number verification workflow evidence. These targets are initial suggestions to be calibrated during pilot design; they close gaps G-06 and G-07.

Go/No-Go Checklist reference.. For board meetings, use the fixed one-page decision card in Appendix A.4 so review panels can evaluate artifacts, thresholds, critical risks, and out-of-scope acknowledgements on a single page.

17.9 Pilot results reporting format (submission template)

To reduce ambiguity in institutional reviews, each pilot report should submit results in a fixed format with raw-artifact references.

KPI	Target	Required pilot evidence
Message delivery success	$\geq 95\%$ indoor/urban; $\geq 85\%$ rubble-like	Trial logs with message IDs, ACK outcomes, and timeout outcomes per scenario.
Text latency	$P95 \leq 5$ s	End-to-end send/receive timestamps with per-device clock handling notes.
Voice envelope	Median one-way latency ≤ 150 ms; packet loss $\leq 3\%$	Call session traces with jitter buffer settings and packet-loss calculations.
Power impact	Within mission-defined limits	Battery drain comparison: standby discovery vs. active transport/voice sessions, including hybrid standby with internet registration and push listeners active (hybrid pilots); attach paired idle-vs-standby traces as supplemental evidence and close the gate on release-equivalent builds.
Responder workflow	100% scripted pass rate	Signed validation checklist for SOS, role proof, secure message exchange, and (if enabled) authorized mesh chat activation.
Public-channel abuse containment	$\geq 99\%$ malformed/out-of-window packets dropped before relay; 100% unverified responder announcements downgraded to untrusted rendering.	Reason-coded drop counters + trusted-announcement positive/negative validation logs.
Internet-path delivery (hybrid pilots)	$\geq 99\%$ relay delivery within 10 s on reference networks; 100% queue recovery on reconnect	Relay traces with envelope IDs, acknowledgement outcomes, and reconnect-flush logs per scenario (gap G-06).
Push wake-up reliability (hybrid pilots)	$\geq 95\%$ background delivery within 60 s across the approved device matrix	Per-device-class wake matrix covering OEM power policies and iOS background states (gap G-06).
Transport handover continuity (hybrid pilots)	100% conversation continuity across Bluetooth/internet	Handover logs with per-message transport labels and deduplication

Field	Required format
Pilot run ID	Unique identifier (site-date-team-run) and document revision hash.
Build provenance	App version, git commit, ABI, signing profile, and feature flags enabled.
Device matrix	Device model, OS version, battery health/charge state, radios enabled.
Scenario definition	Environment class, obstacle profile, distance bins, RSSI bins, motion assumptions, interference notes.
Message results	Total attempts, success count, timeout count, median and P95 latency by scenario.
Voice results	Call setup time, one-way latency distribution, packet-loss estimate, dropout count.
Power results	Baseline drain vs active-mode drain, measurement window, charging conditions.
Failure taxonomy	Categorized root-cause counts (permission, pairing, handshake, transport, decode, persistence).
Public-mesh abuse counters	Reason-coded counts: invalid timestamp, hop-limit stop, inbound-rate drop, oversize drop, duplicate drop, and unverified-announcement downgrade.
Evidence attachment list	Log paths, checksum(s), anonymization method, and reviewer access path.

17.10 Claim-Evidence Matrix (institutional review artifact)

Claims used for Go/No-Go decisions should be explicitly traceable to evidence artifacts.

Claim ID	Claim	Evidence artifact	State
C-01	The offline incident path remains device-to-device with no cloud dependency, and internet-path failure leaves it untouched.	Architecture boundary evidence + local packet-path code review artifacts (no cloud dependency in the offline path; bounded, backoff-limited internet bring-up).	Local
C-02	Rescue interaction is role-gated before secure chat activation.	Role-gate implementation evidence + Rescue workflow gate definitions.	Local

Claim ID	Claim	Evidence artifact	State
C-03	Designated secure messaging paths preserve confidentiality and integrity in transit.	Crypto configuration snapshot + explicit Public GATT boundary note + critical-path crypto test-gap register and expansion plan.	Review
C-04	Voice path stays within the declared voice envelope.	Adjacent-device baseline, two-wall short-range call logs, and debug packet-age instrumentation summary (March 15, 2026) plus scenario traces with synchronized median/P95 latency, loss, and dropout data.	Pending
C-05	Battery impact stays within mission thresholds.	Pixel 7a paired idle-vs-standby trace comparison (March 15, 2026) plus controlled baseline-vs-active drain report with repeat runs.	Review
C-06	Public-channel abuse controls and trusted-announcement gates are enforced.	Reason-coded drop counters; validation logs (valid/expired/wrong-role/bad-signature); operator UI downgrade evidence.	Pending
C-07	The web control plane holds no plaintext incident content while supporting authority telemetry, Settings governance, Crisis Sentinel, workflow settings, automation controls, and ciphertext-only operator messaging.	Reviewed <code>crisis-connect-web</code> : guarded layouts, server sessions, overview and Settings APIs, Crisis Sentinel provider/RAG/release routes, workflow settings tabs, AI/workflow APIs, storage driver, Docker, Helm, SSO paths, cloud rules (server-only key collections, deny-by-default call rooms), and deployment guide.	Local

Claim ID	Claim	Evidence artifact	State
C-08	Internet-path message content is end-to-end encrypted with cryptographic sender authenticity; relay infrastructure can route but not decrypt or forge.	Envelope v2/v3 source review, cross-platform golden-vector suites, relay/rules review (ciphertext-only collections, server-stamped sender), plus pending field delivery traces.	Review
C-09	Current builds provide forward secrecy and downgrade resistance on internet sessions with user-verifiable safety numbers.	libsignal integration review (PQXDH + Double Ratchet), anti-downgrade pin behavior, TOFU/key-change UX evidence, prekey lifecycle tests; pending scenario evidence on representative devices.	Review
C-10	Relay-side ciphertext retention is bounded (delete-on-acknowledgement plus TTL purge).	Backend function/source review and pending instrumented verification of deletion behavior in a pilot environment.	Pending
C-11	SFU group calls expose only MLS-encrypted media frames and opaque handshake state to infrastructure, and the capability is not claimed on platforms where it is feature-gated off.	SFU proxy and rules review, MLS worker source review, iOS feature-gate verification; pending multi-party interop validation evidence.	Pending

17.11 Reviewer-visible open gaps (July 17, 2026)

Gap ID	Linked claim	Current state (v1.1.7 / July 17, 2026)	Closure artifact expectation
G-01	C-04 (voice envelope)	March 15, 2026 adjacent-device and two-wall short-range call log sets plus debug packet-age instrumentation runs are still attached, but they were captured on 0.9.9-debug , one-way latency evidence remains narrow, and the packet still does not represent range-binned or explicitly degraded-RF scenarios for the current v1.1.7 source line. The new GATT and WebRTC call paths have no envelope evidence at all yet.	Scenario-wise call traces with synchronized median/P95 latency, loss estimate, and dropout logs on release-equivalent current-line builds, covering RFCOMM, GATT (cross-platform), and WebRTC paths.
G-02	C-05 (battery)	A paired Pixel 7a idle-vs-standby trace set is attached as supplemental evidence, but it comes from a debug, AC-powered run on a single device and there is still no release-equivalent baseline-vs-active comparison artifact in the packet.	Repeatable baseline-vs-active drain report with device/environment controls, repeat runs, unplugged conditions, and release-equivalent builds.
G-03	C-06 / T-08 (public-channel abuse)	Source controls and parameter locks are documented, but packet-level stress counters and trusted-announcement downgrade proofs are still pending field/instrumented execution.	PM-01..PM-05 and TA-01..TA-06 run outputs (<code>summary_counts.txt</code> , UI/log attachments).
G-04	C-03 support depth (crypto tests)	The repository still lacks dedicated automated vectors for ECDH/HKDF derivation and replay-cache/rate-limit negative paths on the current v1.1.7 line.	Added crypto-focused unit/integration tests plus reproducible CI run records.
G-05	C-07 (web control-plane completeness)	The main dashboard and selected Settings/Crisis Sentinel/workflow APIs are on-prem aware, and July 2026 closed the E2E-test, rate-limit, scheduler-reliability, and secret-handling gaps. Remaining gaps include Firebase-client account/profile, notifications, organization-role editing, AI chat persistence, workflow list/run realtime reads, some security-log reads, public basemap tiles, and missing backup/restore artifacts.	Server-backed or explicitly excluded scope for required Settings/AI/workflow screens, workflow settings policy proof, self-hosted map tiles, broader role/panel/messaging E2E tests, Crisis Sentinel/RAG/model-release evidence, AI/MCP credential governance, webhook negative tests, a tested backup/restore runbook, and a signed deployment packet.

17.12 Compliance mapping (control owner vs required artifact)

Institutional review should map each control family to a named owner and a concrete evidence artifact.

Control family	Control owner	Required review artifact
Transport boundary assurance	Mobile engineering lead	Packet-path evidence showing the offline incident path remains local device-to-device with no cloud dependency, and that the internet path relays end-to-end encrypted (ciphertext-only) envelopes with plaintext confined to end-user devices.
Internet-plane relay governance	Backend / platform engineering lead	Relay retention verification method and results (delete-on-acknowledgement, TTL purge), push wake-up measurement plan and pass-rate matrix, TURN/push/relay/STUN provider dependency declaration with hosting-region statement, and backend attestation-gate configuration snapshot.
Responder trust lifecycle	Security / PKI lead	Issuance policy, authority key management record, and offline verification policy snapshot.
Cryptographic profile integrity	Security engineering lead	Versioned cryptographic profile sheet (algorithms, key sizes, nonce/replay policy, downgrade handling).
Operational readiness	Field operations lead	Signed drill checklist and failure-handling playbook for SOS and responder workflow.
Privacy and retention controls	Privacy / compliance lead	Data retention policy, log minimization specification, and deletion workflow evidence.
Public-channel abuse governance	Operations security lead	Declared Public GATT parameter freeze record, abuse-counter export, and trusted-announcement SOP/test report.

17.13 Pilot risk register (operational and security)

Risk ID	Trigger condition	Impact	Required mitigation / gate control
R-01	Aggressive OEM power policy suppresses BLE scan/advertise continuity.	Reduced discoverability and delayed delivery.	Device-policy whitelist plus per-OEM validation evidence before deployment approval.
R-02	Responder credentials are not rotated or revoked on schedule.	Elevated impersonation or stale-credential risk.	Short-lived certificates, periodic denylist distribution, and audited rollover procedure.
R-03	Voice profile stays in high-rate mode during RF degradation.	Call instability, queue growth, and battery drain spikes.	Mandatory adaptive profile policy with trace logs for mode-switch decisions.
R-04	Pilot logs contain linkable personal metadata.	Privacy and regulatory exposure.	Pseudonymous test IDs, retention bounds, and controlled log-access workflow.
R-05	Post-release regressions emerge after v1.1.7 baseline changes (feature modularization, mesh/crisis-link/runtime refactors).	Reduced institutional confidence and delayed deployment readiness.	Maintain CI trend monitoring, pin reproducible test environments, and require rapid rollback/isolation policy for newly failing suites before Go/No-Go.
R-06	Legacy protocol regressions reintroduce key-carry fields into handshake traffic.	Key-management boundary erosion and downgrade risk across mixed-version peers.	Freeze HSK_REQ challenge ivB64 cipherB64 contract (challenge+IV+cipher only), ignore legacy trailing fields for trust decisions, and enforce protocol regression tests before release approval.

Risk ID	Trigger condition	Impact	Required mitigation / gate control
R-07	Public GATT channel is used for sustained spam/rumor flooding or untrusted mass announcements.	Priority instructions are crowded out, and misinformation spreads faster than verification workflows.	Enforce declared Public GATT bounds (hop/age/rate/size/queue), maintain mute/report workflow, and require responder-only announcement template with verifier-visible identity marker.
R-08	Operators or civilians treat the internet path as guaranteed reach during an incident.	Missed communications when infrastructure degrades; misplaced operational trust in push delivery.	Training and UI language keep Bluetooth as the incident-default mental model; pilots measure push wake-up and relay delivery rates and publish them as an operating envelope, not a guarantee.
R-09	Default-on anonymous registration and presence signals create unreviewed personal-data exposure under local data-protection law (e.g., KVKK/GDPR).	Regulatory findings, deployment delays, or forced feature changes after rollout.	Complete the pending legal disclosure review before public-facing claims; document the registration boundary, retention, and processor inventory in the deployment packet; offer QR-only/offline-dominant profiles for sensitive deployments.
R-10	Third-party relay/TURN/push providers change terms, throttle, or fail regionally mid-operation.	Internet-path calls and messaging degrade or fail while the offline path continues.	Declare provider dependencies in procurement, monitor fallback rates (STUN-only, store-and-forward), and exercise Bluetooth-bridge fallback for authority channels in drills.

17.14 Current validation snapshot (as of March 31, 2026)

The March 31, 2026 snapshot below was captured during the Android 1.0.0 review pass and is retained as the last frozen QA inventory. It predates the 1.1.x hybrid line: since then the test surface has grown alongside the new capabilities (cross-platform crypto golden

vectors, Signal-session suites, call-protocol and SPAKE2 tests on both mobile platforms, and Playwright end-to-end suites on the web line), and a refreshed frozen inventory for v1.1.7 is an open release-packet item. This subsection summarizes the source-visible validation scope and the artifact bundle retained for release handoff as of the snapshot date.

- 38 unit-test source files are currently present under `app/src/test`.
- 197 declared `@Test` methods are present in the app unit-test source tree.
- Local QA artifacts for this revision are retained under `tmp/evidence_20260331` for engineering handoff and release-packet preparation.

Unit-test source area	Current signal
OfflineRegionRepositoryImplTest, StartDownloadUseCaseTest, OfflineMapViewModelTest ChatEncryptionSetupScreenTest, QRHelpersTest, RoleCertificateTest AesGcmTest, ChunkerTest, TransferIdsTest, VoiceStreamConstantsTest	Offline-map repository / use-case / view-model coverage remains present in source and should be exercised in the release QA packet for Android v1.1.7. QR onboarding and responder-certificate validation coverage remains present in source. Basic AES-GCM / chunking / transfer-ID / voice-constant guardrails remain covered in source; this still does not by itself close ECDH / HKDF / replay stress evidence gates.
MessageDaoTransactionTest, ContactTrustStateTest, GattMeshChatStoreTest, ProfileImageStorageTest BleChatEnvelopeTest, BlePeerIdentityUtilsTest, BleImagePayloadTest, BleVoicePayloadTest, File / Image stream constant tests	Persistence, contact-trust state, local storage, and mesh-chat store coverage remains present in source. BLE envelope parsing, peer-identity normalization, and media payload guardrails remain covered in source.
CrisisLinkQueueCodecTest, CrisisLinkSyncDecisionEngineTest, ServiceDecisionFlowIntegrationTest GattMeshCommandDeciderTest, ChatFormattersTest, ChatLocationUtilsTest, ChatScreenOrientationTest, RescueScreenViewModelTest, whistle / tool tests	Crisis Link queue / sync / service-decision coverage remains represented in source and stays within the Android v1.1.7 release-review scope. Mesh command filtering plus chat / tool UI logic coverage remains present in source and should accompany the release QA packet for this baseline.

Critical-path coverage caveat. The repository still lacks dedicated automated vectors for ECDH / HKDF derivation correctness, responder role-proof nonce-mismatch negative paths, and Public GATT replay / rate-limit stress behavior.

Post-pass governance note. Pilot packets SHOULD include CI history, reproducibility metadata, and scenario-level field evidence for transport, Rescue Mode, and optional mesh workflows (authorized secure responder mesh and Public GATT general channel where enabled).

17.15 Local evidence bundle snapshot (March 31, 2026)

This subsection records reproducible local artifacts collected during the March 31, 2026 (Android 1.0.0) review pass and retained for this edition. Source-provenance, build-config, and unit-test summary artifacts were last regenerated on March 31, 2026. Previously frozen Public Mesh artifacts from March 3, 2026 are retained where the reviewed v1.1.7 code paths still match their declared controls.

Artifact	Artifact reference (ID + relative path)	Observed value / revision note
Source provenance snapshot	E-12 (tmp/evidence_20260331/source_provenance.txt)	Branch <code>main</code> ; subject “Crisis Connect Alpha 1.0.0(32)”; commit time <code>2026-03-26T23:08:18+03:00</code> ; working tree remains dirty because the local workspace has further uncommitted edits.
Build config snapshot	E-13 (tmp/evidence_20260331/build_config_snapshot.txt)	<code>app/build.gradle.kts</code> declared <code>versionName 1.0.0</code> , <code>versionCode 33</code> , <code>minSdk 24</code> , <code>targetSdk 35</code> , <code>compileSdk 35</code> , and dynamic feature <code>:feature_rescue</code> at snapshot time. Release packaging for the Android 1.0.0 review pass was generated from this line; a refreshed build-config snapshot for the current v1.1.7 (versionCode 57) line is an open release-packet item.
Public mesh gating snapshot	E-14 (tmp/evidence_20260303/public_mesh_gating.txt)	March 3 snapshot retained; March 31 code review still confirms Public mesh default <code>false</code> , explicit settings toggle activation, and blocked general-chat send path when Public mesh is disabled.
Public GATT packet-format snapshot	E-15 (tmp/evidence_20260303/public_gatt_packet_format.txt)	March 3 snapshot retained; March 31 code review still confirms JSON core fields (<code>id</code> , <code>sender</code> , <code>timestamp</code> , <code>hop</code> , <code>protocol</code>) plus encrypted chat envelope fields (<code>keyId</code> , <code>iv</code> , <code>cipher</code>) in the non-confidential Public GATT profile.
Public-mesh abuse counters snapshot	E-17 (tmp/evidence_20260303/public_mesh_abuse_counters.json)	March 3 source-conformance snapshot retained; current v1.1.7 code review still shows declared <code>hop/age/rate/size/queue</code> bounds, but packet-level field counters remain pending stress-run capture.

Artifact	Artifact reference (ID + relative path)	Observed value / revision note
TA validation report	E-18 (trusted-announcement report)	Pending-field validation matrix for downgrade proof.
T-08 closure protocol	E-19 (tmp/evidence_20260303/t08_closure_protocol.md)	Step-by-step scenario matrix and closure criteria for PM-01..PM-05 and TA-01..TA-06 evidence runs.
T-08 command-set script	E-20 (tmp/evidence_20260303/collect_t08_evidence.sh)	Executable start/stop/summarize script for log capture and reason-count extraction during field/instrumented runs.
Unit-test run summary	E-16 (tmp/evidence_20260331/unit_test_summary.txt)	Local QA summary retained for engineering handoff; source inventory records 38 test files and 197 declared tests in the app unit-test tree for the reviewed Android baseline.
HTML test report	E-16-HTML (internal artifact-store reference)	Attach the current CI or local execution export used for the final release QA packet when available.

17.16 iOS deployment-baseline kit (April 3, 2026)

To accelerate promotion of the iOS 1.1.2 line from implementation baseline to deployment baseline, the workspace now includes a dedicated kit under `tmp/evidence_20260403_ios_baseline`. These files are not deployment evidence by themselves; they are structured operator and reviewer artefacts intended to force consistency across release control, scenario execution, device capture, and claim-to-evidence mapping. One provenance note: the kit's recorded version label ("1.0.0 (1)") predates the repository history — the app target already carried marketing version 1.0.3 at the initial commit, and the line later advanced directly to 1.1.2 with no intermediate 1.1.0/1.1.1 releases — so reviewers reconciling kit artifacts against the repository should expect this offset.

The intended workflow is deliberate: freeze the release candidate first, execute scenarios against the frozen build, record every device and condition in the device matrix, map surviving claims to concrete artefacts, and only then issue an assurance recommendation. If that order is reversed, reviewers typically end up with inconsistent build IDs, unverifiable screenshots, and claims that are broader than the retained evidence. Because a signed iOS archive now exists, the next professional bottleneck is no longer release packaging but execution discipline. The playbook and session checklist above are therefore intended to turn the remaining Apple-device runs into a reviewer-auditable program rather than an informal smoke pass.

Kit artifact	Purpose
tmp/evidence_20260403_ios_baseline/README.md	Operator overview for how to assemble the iOS deployment-baseline packet in the correct order.
tmp/evidence_20260403_ios_baseline/signed_archive_snapshot_2026-04-03.md	Current signed-archive snapshot covering archive path, signing identity, checksums, and Apple-upload traceability for the present iOS candidate.
tmp/evidence_20260403_ios_baseline/release_manifest_template.md	Canonical release-candidate freeze sheet for build identity, signing, dependency, provenance, and export details.
tmp/evidence_20260403_ios_baseline/assurance_report_template.md	Reviewer-grade summary for automated tests, manual scenarios, open defects, waivers, and promotion recommendation.
tmp/evidence_20260403_ios_baseline/device_matrix_template.csv	Structured Apple-hardware matrix covering model class, OS version, LiDAR capability, permissions, and runtime conditions.
tmp/evidence_20260403_ios_baseline/scenario_matrix_template.md	Repeatable run ledger for QR/BLE, voice, Rescue, live location, offline-map, and LiDAR/Night Vision scenarios.
tmp/evidence_20260403_ios_baseline/field_validation_playbook_2026-04-04.md	Reviewer-grade execution playbook defining staffing, execution waves, evidence minimums, stop conditions, and promotion interpretation for the current signed iOS candidate.
tmp/evidence_20260403_ios_baseline/session_execution_checklist_2026-04-04.md	Concise session-floor checklist for validation leads and evidence custodians during live device runs.
tmp/evidence_20260403_ios_baseline/physical_ipad_deployment_snapshot_2026-04-04.md	First physical iPad deployment snapshot covering device identity, device-targeted build, install, launch, and process evidence for the current candidate.
tmp/evidence_20260403_ios_baseline/physical_ipad_manual_workflow_snapshot_2026-04-05.md	First manual iPad workflow snapshot covering tab-surface smoke, offline-map primary flow, and a preliminary Night Vision runtime observation on the current candidate.
tmp/evidence_20260403_ios_baseline/ios_android_basic_interop_snapshot_2026-04-05.md	First basic mixed-platform interoperability snapshot covering iPad-to-Android pairing and successful message exchange on the current candidate.
tmp/evidence_20260403_ios_baseline/claim_evidence_map_template.md	Whitepaper-claim to artefact mapping sheet used to narrow or validate iOS-facing statements before promotion.
tmp/evidence_20260403_ios_baseline/known_limitations_template.md	Residual-risk and unsupported-behavior sheet for reviewers and institutional operators.

17.17 Supplemental local evidence snapshot (March 15, 2026)

This subsection records the standby power and supplemental voice notes added after the March 3, 2026 packet freeze. The values below are cross-checked against the raw exports used during this revision pass and should be treated as supplemental, reviewer-visible evidence rather than as a replacement for the full pilot artifact set. These traces predate the current v1.1.7 source baseline and were collected on 0.9.9-debug builds, so they remain directional rather than release-equivalent closure artifacts.

Artifact	Artifact reference (ID + relative path)	Observed value (March 15, 2026)
Pixel 7a standby power comparison summary	E-21 (tmp/evidence_20260315/standby_power_trace_summary.txt)	Baseline idle 233.6 mW vs app standby 247.7 mW over matched 300 s windows; incremental standby overhead 14.1 mW; debug, AC-powered run, supplemental evidence only, not gate closure.
Pixel 7a baseline idle raw trace	E-22 (tmp/evidence_20260315/pixel7a_idle_baseline.trace)	Raw Perfetto export for app force-stopped baseline window.
Pixel 7a app standby raw trace	E-23 (tmp/evidence_20260315/pixel7a_app_standby.trace)	Raw Perfetto export for background-ready app standby window.
Adjacent-device voice baseline summary	E-24 (tmp/evidence_20260315/voice_gate_run_01/voice_call_baseline_summary.txt)	Two-device adjacent RFCOMM call run; answered teardown on both devices after approximately 126 s with zero observed loss/underrun/CRC-drop counters in periodic playback logs; supplemental clean-baseline evidence only, not voice-gate closure.
Two-wall voice continuity summary	E-27 (tmp/evidence_20260315/voice_gate_run_02/voice_call_two_walls_summary.txt)	Two-device short-range call with two walls; answered teardown on both devices after approximately 212 s; zero observed loss/underrun/CRC-drop counters but materially higher sustained jitter-buffer depth than the adjacent-device run; supplemental continuity evidence only, not voice-gate closure.

Artifact	Artifact reference (ID + relative path)	Observed value (March 15, 2026)
Voice latency instrumentation summary	E-30 (tmp/evidence_20260315/voice_latency_run_02/voice_latency_instrumentation_summary.txt)	Two-device debug instrumentation call; answered teardown on both devices after approximately 172 s with zero observed underrun/loss/CRC-drop counters; receiver-observed packet-age summaries are now attached, but the values are dominated by handset clock skew and therefore support instrumentation validation rather than gate-closing absolute one-way latency claims.
Voice clock-adjusted latency summary	E-33 (tmp/evidence_20260315/voice_latency_run_05/voice_clock_adjusted_latency_summary.txt)	Two-device debug call with monotonic clock-sync calibration; answered teardown on both devices after approximately 64–65 s; clock-adjusted packet-age median 16–18 ms, P95 99–120 ms, max 299–356 ms, syncDiscarded 0 on both devices; strong supplemental latency evidence, but still not full voice-gate closure.

18 Use Cases

18.1 Remote & Wild (Off-Grid Exploration)

In off-grid settings, the primary objective is group safety and coordination in areas with no cellular coverage. The system supports local messaging and location sharing within the group.

18.2 Hybrid Continuity (Degraded Infrastructure)

In partial-outage and recovery phases — overloaded base stations after a quake, rolling blackouts, or intermittent backhaul — the hybrid queue keeps one conversation alive across transports: messages ride Bluetooth when peers are near, drain over the low-bandwidth E2E internet path when a connectivity window opens (template messages ease network load), and family or team members outside the disaster zone remain reachable on the same contact entry. SOS events additionally reach authority dashboards and top contacts while connectivity lasts, without ever gating the local BLE beacon on it.

18.3 Infrastructure Collapse (Earthquake Response)

In earthquake and building-collapse scenarios, Rescue Mode supports local SOS signaling and authenticated responder communication during proximity-based search sweeps.[10, 11] RSSI can assist search teams as a coarse indicator, but should be treated as a heuristic rather than a precise ranging tool.[8, 9]

19 Limitations and Safety Notes

- Bluetooth is proximity-based; coverage is fundamentally limited and environment dependent.
- The system is designed for short-range operational support in local sectors and does not guarantee reach to every victim under all obstruction conditions.
- Default civilian flows rely on direct local links; v1.1.7 mesh modes require compatible radio hardware, explicit operator enablement, and local-radio proximity.
- Public GATT mesh is an opt-in broad-coordination channel and is disabled by default; only users who explicitly enable “Advanced Settings > Public mesh mode” can use general mesh chat.
- Public GATT mesh is not recommended as an always-on setting for every user profile; enable it when broad nearby coordination is needed.
- Confidential operational content should remain on secure, role-gated responder paths.
- Unbounded unmanaged relay / store-and-forward behavior remains out of scope.
- The internet path depends on surviving third-party infrastructure (relay backend, push services, TURN); it extends reach when networks work and disappears with them. It must never be presented as a delivery guarantee during infrastructure loss.
- Internet-path confidentiality is end-to-end by design, but routing metadata (identifiers, timing, push tokens) remains visible to backend infrastructure and should be considered in sensitive deployments.
- SFU group calling is not available on iOS in the current line (feature-gated pending interop validation), and iOS-originated screen share is not yet implemented.
- Performance varies across devices, OS versions, and power-management policies.
- Crisis Connect does not replace official emergency services; users should attempt to contact the local emergency number (for example, 112, 911, or the nationally designated line) when possible.

20 Conclusion

What the system solves. Crisis Connect defines a resilient, offline-first communication layer for disaster conditions: a cloud-free local path when wide-area infrastructure is unavailable, and an end-to-end encrypted internet extension of the same conversations while infrastructure survives, with responder authenticity and endpoint-only plaintext preserved on every designated secure path.

What the pilot must prove. Institutional confidence depends on measured evidence, not architecture claims. Pilot execution should therefore report delivery success, latency, voice stability, power impact, and responder workflow success against the acceptance gates in the Evaluation Plan — now on both planes: the offline gates carried forward from earlier editions, and the new internet-plane gates (relay delivery, push wake-up, transport handover, retention verification, forward-secrecy workflow evidence).

Next institutional step. Operating authorities should run controlled field pilots with governance artifacts (credential lifecycle, revocation snapshots, device policy baselines, and training procedures), then authorize phased deployment only for environments where the measured envelope meets mission requirements.

References

- [1] W. Ladd and B. Kaduk, “Rfc 9382: Spake2, a password-authenticated key exchange,” IETF, RFC 9382, 2023. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc9382>
- [2] Bluetooth SIG, “Core specification 6.2 (adopted),” 2025, dOI not assigned by publisher. [Online]. Available: <https://www.bluetooth.com/specifications/specs/core-specification-6-2/>
- [3] —, “Rfcomm 1.2 (adopted),” 2026, dOI not assigned by publisher. [Online]. Available: <https://www.bluetooth.com/specifications/specs/rfcomm-1-2/>
- [4] Google, “Wi-fi aware overview,” 2026, android Developers. DOI not assigned by publisher. [Online]. Available: <https://developer.android.com/develop/connectivity/wifi/wifi-aware>
- [5] —, “Wifiawaremanager,” 2026, android Developers API reference. DOI not assigned by publisher. [Online]. Available: <https://developer.android.com/reference/kotlin/android/net/wifi/aware/WifiAwareManager>
- [6] —, “Scansettings,” 2026, android Developers API reference. DOI not assigned by publisher. [Online]. Available: <https://developer.android.com/reference/android/bluetooth/le/ScanSettings>

- [7] J. Siva, J. Yang, and C. Poellabauer, "Connection-less ble performance evaluation on smartphones," *Procedia Computer Science*, vol. 155, pp. 51–58, 2019. [Online]. Available: <https://doi.org/10.1016/j.procs.2019.08.011>
- [8] R. Ramirez, C.-Y. Huang, C.-A. Liao, P.-T. Lin, H.-W. Lin, and S.-H. Liang, "A practice of ble rssi measurement for indoor positioning," *Sensors*, vol. 21, no. 15, p. 5181, Jul. 2021. [Online]. Available: <https://doi.org/10.3390/s21155181>
- [9] F. Milano, H. da Rocha, M. Laracca, L. Ferrigno, A. Espírito Santo, J. Salvado, and V. Paciello, "Ble-based indoor localization: Analysis of some solutions for performance improvement," *Sensors*, vol. 24, no. 2, p. 376, Jan. 2024. [Online]. Available: <https://doi.org/10.3390/s24020376>
- [10] Y. Tai and T.-T. Yu, "Using smartphones to locate trapped victims in disasters," *Sensors*, vol. 22, no. 19, p. 7502, Oct. 2022. [Online]. Available: <https://doi.org/10.3390/s22197502>
- [11] Y. Maruyama and T. Miyazaki, "Hand-held system to find victims with smartphones in disaster environment," *IEICE Transactions on Communications*, vol. E104.B, no. 4, pp. 455–462, Apr. 2021. [Online]. Available: <https://doi.org/10.1587/transcom.2020EBP3040>
- [12] National Institute of Standards and Technology, "Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac," NIST, NIST Special Publication 800-38D, 2007. [Online]. Available: <https://csrc.nist.gov/pubs/sp/800/38/d/final>
- [13] Google, "Bluetooth low energy," 2026, android Developers. DOI not assigned by publisher. [Online]. Available: <https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview>
- [14] —, "Advertisesettings," 2026, android Developers API reference. DOI not assigned by publisher. [Online]. Available: <https://developer.android.com/reference/android/bluetooth/le/AdvertiseSettings>
- [15] E. Kret and R. Schmidt, "The pqxdh key agreement protocol," Signal protocol specification, 2023, dOI not assigned. [Online]. Available: <https://signal.org/docs/specifications/pqxdh/>
- [16] T. Perrin and M. Marlinspike, "The double ratchet algorithm," Signal protocol specification, 2016, dOI not assigned. [Online]. Available: <https://signal.org/docs/specifications/doubleratchet/>
- [17] H. Alvestrand, "Rfc 8825: Overview: Real-time protocols for browser-based applications (webrtc)," IETF, RFC 8825, 2021. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc8825>

- [18] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon, "Rfc 9420: The messaging layer security (mls) protocol," IETF, RFC 9420, 2023. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc9420>
- [19] National Institute of Standards and Technology, "Digital signature standard (dss)," NIST, Federal Information Processing Standards Publication 186-5, 2023. [Online]. Available: <https://csrc.nist.gov/pubs/fips/186-5/final>
- [20] Certicom Research, "Sec 1: Elliptic curve cryptography, version 2.0," Standards for Efficient Cryptography Group, Tech. Rep., 2009, dOI not assigned. [Online]. Available: <https://www.secg.org/sec1-v2.pdf>
- [21] National Institute of Standards and Technology, "Fips 203: Module-lattice-based key-encapsulation mechanism standard," NIST, FIPS 203, 2024. [Online]. Available: <https://csrc.nist.gov/pubs/fips/203/final>
- [22] H. Krawczyk and P. Eronen, "Rfc 5869: Hmac-based extract-and-expand key derivation function (hkdf)," IETF, RFC 5869, 2010. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc5869>
- [23] J.-M. Valin, K. Vos, T. B. Terriberry, and G. Maxwell, "Rfc 6716: Definition of the opus audio codec," IETF, RFC 6716, 2012. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc6716>
- [24] Google, "Communicate in the background," 2026, android Developers. DOI not assigned by publisher. [Online]. Available: <https://developer.android.com/develop/connectivity/bluetooth/ble/background>
- [25] Auralis Industries, "Crisis connect project website," 2026, dOI not assigned by publisher. [Online]. Available: <https://crisisconnect.network>

A Implementation Status and Roadmap Notes

- The protocol profile in the Reference Communication Architecture section defines the baseline wire format and security safeguards used for pilot validation, and Section 8 defines the internet-plane envelope and session profiles. Release notes should map implementation versions to these profiles.
- Android baseline v1.1.7 documents hybrid transport behavior: the offline Bluetooth path with controlled responder-only Wi-Fi Aware secure mesh (plus media accelerators and PTT) and opt-in Public GATT general channel, and an internet E2E relay drain of the same message queue. Unbounded unmanaged relay/store-and-forward behavior remains out of scope.
- Crisis Link telemetry synchronization and SOS cloud uplink in v1.1.7 are operational metadata only; the internet message path itself is ciphertext-only relay, and plaintext emergency content MUST remain endpoint-only.

- Managed deployments may additionally use the enterprise SSO bridge with profile synchronization to a configurable institutional backend (mobile-sync); this is an administrative identity/profile channel, not a message path.
- Crisis Connect Web **0.1.0** is documented as the reviewed operations dashboard/control-plane baseline. It supports cloud/on-prem modes, server-side sessions, Postgres/Keycloak storage/auth, Docker Compose evaluation, Helm deployment, Settings governance APIs, Crisis Sentinel provider/RAG/model-release controls, AI workflow policy controls with cron scheduling and MCP exposure, Telegram/Slack assistant channels, E2E operator messaging and SFU/MLS group calls, ES256 license checks, signed update manifests, and optional installation health reporting; plaintext incident content MUST remain outside the control plane.
- The iOS line covers onboarding, QR/BLE pairing, BLE point-to-point flows, the full internet messaging/calling plane in Android parity, cross-platform GATT voice calls, Night Vision, offline field tooling, Crisis Sentinel, 19-language localization, and role-gated Rescue workflows under CI/TestFlight governance. It remains an iOS **1.1.2** parity baseline pending promotion testing, with the newest feature branch not yet merged to the release branch.
- Version alignment requirement: this document is published as whitepaper **2.0.0** while the reviewed Android implementation source baseline is currently **v1.1.7** (versionCode 57) and the parallel iOS implementation baseline is currently **1.1.2**. Institutional packets should include an explicit traceability matrix between document claims and app build identifiers, and for the web line should bind to image tag and build SHA.

A.1 Release-package checklist (external publication)

- Include finalized protocol, Rescue Mode, and trust-architecture diagrams with version-stamped legends.
- Attach claim-evidence and compliance-mapping artifacts signed by designated control owners.
- Include pilot raw-artifact manifest (checksums, anonymization notes, and reviewer access method).
- Include cryptographic profile sheet with algorithm/parameter versions and key-rotation policy reference.
- Include the appendix one-page board card (Appendix A.4) in decision-meeting review packets.
- Include Public-mesh abuse counter export and trusted-announcement validation report when open-channel mode is in pilot scope.
- Include T-08 closure protocol and command-set artifacts (`t08_closure_protocol.md`, `collect_t08_evidence.sh`) when C-06/T-08 is pending closure.
- Include reviewer access links (APK access workflow, technical walkthrough, and source provenance commit links).

- Include the web control-plane deployment packet when dashboard/on-prem review is in scope: web source identifier, image tag, build SHA, `Dockerfile`, `docker-compose.yml`, Helm values, migration record, SSO provider manifest, role/panel authorization test evidence, Settings/AI/workflow route coverage, Crisis Sentinel provider/RAG/model-release evidence, AI provider and MCP credential inventory, workflow settings policy snapshot, license/update public keys, health-reporting configuration, rollback/upgrade runbook, managed-secret inventory, and sanitized environment template.
- If iOS deployment-baseline promotion is sought, include the full iOS deployment-baseline packet defined in Section 12.3 (release manifest, signed-archive snapshot, assurance report, device matrix, scenario ledger, field-validation playbook, claim-to-evidence map, and known-limitations sheet).
- Pre-populate the iOS template kit under `tmp/evidence_20260403_ios_baseline` before reviewer dry-runs so missing artefacts are discovered before formal Go/No-Go review.
- Include the internet-plane assurance set when hybrid features are in pilot scope: envelope/session golden-vector test exports, relay and rules review notes (ciphertext-only collections, server-only key stores), push wake-up and delivery measurement plans, relay retention verification method, TURN/push/relay provider dependency declaration, safety-number verification SOP, and the SFU/MLS gate status per platform.
- Include the privacy inventory for default-on registration, presence signals, analytics event schema, and directory opt-in flows, aligned to the local data-protection review (KVKK/GDPR) status.
- Include a deployment safety note summarizing scope boundaries and non-goals.
- Include explicit emergency-service escalation guidance.

A.2 Reviewer access links (demo / APK / source)

Artifact class	Reviewer access link / identifier
Android release artifact (v1.1.7 reviewer target)	Owner-distributed APK/AAB package aligned to the Android v1.1.7 (versionCode 57) store launch workflow.
Release coordination workflow	Owner-managed reviewer and launch distribution process.
Private beta APK artifact (v1.1.7 reviewer target)	Owner-distributed secure channel (access-controlled); reviewer artifact should be generated from the current source baseline before distribution.
Release artifact metadata	Internal release manifest and signing outputs generated from the Android v1.1.7 release pipeline.

Artifact class	Reviewer access link / identifier
iOS signed archive evidence (1.1.2 reviewer target)	Signed Xcode Organizer archive with successful Apple-upload metadata retained through the owner-managed reviewer workflow; reviewer-visible snapshot and supporting packet artefacts are indexed under <code>tmp/evidence_20260403_ios_baseline</code> .
Technical walkthrough material	Internal demo deck, walkthrough notes, or approved launch collateral.
Public download landing page	https://crisisconnect.network/download
Google Play Store listing (Android)	https://play.google.com/store/apps/details?id=com.auralis.crisisconnect
Apple App Store listing (iOS / iPadOS)	https://apps.apple.com/us/app/crisis-connect-offline-chat/id6759731195
Public source repository	https://github.com/emirhan-duman/Crisis-Connect
Reviewed web operations workspace	Local <code>crisis-connect-web</code> workspace reviewed for Crisis Connect Web 0.1.0; reviewer packet should include sanitized source/deployment artefacts only: <code>package.json</code> , <code>Dockerfile</code> , <code>docker-compose.yml</code> , <code>DEPLOY.md</code> , <code>docs/agency-panels-production-runbook.md</code> , <code>docs/onprem-update-management.md</code> , <code>helm/crisis-connect</code> , migration SQL, dashboard routes, Settings routes and admin APIs, /ai assistant routes, Crisis Sentinel provider/knowledge/model-release routes, workflow builder/settings/execute/generate/agent/webhook routes, workflow Permissions/Defaults/Models/Credentials/Versioning policy snapshot, node-catalog and dry-run evidence, auth/session routes, storage-driver code, enterprise SSO paths, license/update/health routes, Firestore/Storage rules where cloud mode is in scope, and a production values/template set with AI provider, RAG, MCP, workflow credential, and model-artifact secrets redacted. Local secrets, logs, heap snapshots, dependency backups, and private keys MUST NOT be included.

Artifact class	Reviewer access link / identifier
Standalone iOS codebase status	Standalone SwiftUI iOS 1.1.2 codebase (bundle <code>com.auralis.crisisconnect</code> ; iOS 17+; iPhone/iPad) is carried as the store-listed parity implementation baseline under Codemagic CI/TestFlight governance; it includes onboarding, QR/BLE trust bootstrap, BLE point-to-point flows, the internet E2E messaging/calling plane (v2/v3 profiles, CallKit/PushKit), cross-platform GATT voice calls, authority channel messaging with Bluetooth bridge, Crisis Sentinel, LiDAR-guided Night Vision, offline field tooling, 19-language localization, and role-gated Rescue workflows. The NaN/Wi-Fi Aware path is scoped to authorized Rescue use and remains platform-gated; SFU group calls are feature-gated off pending interop validation; the newest feature branch has not yet passed through the release pipeline; and the app line is not yet part of validated institutional deployment-baseline claims.
Owner-controlled internal Android repository record	Available through owner-managed reviewer workflow (access-controlled).
Owner-controlled local workspace evidence	Retained by the document owner and furnished only during approved technical audit workflows.

Publication note (as of July 17, 2026). Distribution of *this document* and its reviewer evidence packets remains approval-based while launch sequencing and reviewer packaging continue; the Android and iOS applications themselves are publicly available through the store listings above. Reviewer packets should reference the current repository provenance, the Android v1.1.7 (versionCode 57) store line together with the March 31, 2026 offline-path review set it inherits, the iOS 1.1.2 parity line with its April 5, 2026 evidence anchor and current CI/TestFlight posture, and the Crisis Connect Web 0.1.0 control-plane review updated in this edition for the July 2026 capability expansion. New hybrid-plane claims in this edition are source-review claims; their field-evidence gates are tracked as G-06 through G-09.

A.3 Crypto Profile Sheet (reviewer-ready sample)

This compact sheet can be attached directly to procurement or institutional review packets.

Field	Declared value	Evidence / provenance
Profile identifier	CC-CRYPTO-PIL0T-2.0	Cryptographic profile lock + this appendix revision.
Whitepaper binding	2.0.0, revision date July 17, 2026.	Title-page governance table + revision history.
Android build label under review	v1.1.7 source baseline (versionCode 57), distributed through the public store workflow.	Current app/build.gradle.kts + release provenance.
Contact AEAD profile	AES-256-GCM, 96-bit IV, 128-bit tag.	Cryptographic profile lock.
Internet envelope (v2)	Authenticated ECIES: P-256 ECDH (ephemeral-static + static-static), HKDF-SHA-256, AES-256-GCM with sender / recipient / conversation AAD binding.	Section 8 + cross-platform golden vectors.
Internet sessions (v3)	Signal protocol via libsignal: PQXDH (Kyber) + Double Ratchet, TOFU pinning, anti-downgrade enforcement, numeric-fingerprint safety numbers.	Section 8 + client repository test suites.
SPAKE2 bootstrap	RFC 9382 P-256 / SHA-256 / HKDF / HMAC profile pinned to RFC test vectors; AES-256-GCM identity exchange under derived key.	Contact-bootstrap subsection + client test suites.
Group-call media	Per-frame AEAD under OpenMLS 0.7.x group key (SFU path); VP9 pinned across platforms.	Section 8 + shared MLS core.
Responder key agreement	ECDH P-256 + HKDF-SHA-256 transcript-bound derivation.	Cryptographic profile lock + cited standards (nist-dss, hkdf).
Responder authentication	ECDSA P-256 verification against pinned authority trust anchor.	Responder authentication and role-proof gate.
Nonce / key-lifetime governance	CSPRNG IV per packet; bounded packets-per-key; mandatory refresh on restart or re-handshake.	Nonce safeguards + cryptographic profile controls.

Field	Declared value	Evidence / provenance
Replay / duplicate controls	Receiver replay cache (sender, UUID, TTL) + persistence-layer UUID idempotence.	Wire-format safeguards and cryptographic profile controls.
Bluetooth socket/security assumption	RFCOMM compatibility may use secure or insecure socket paths; pairing + post-pairing challenge-response gate is required for contact activation, and confidential-path guarantees rely on app-layer cryptography.	Bluetooth assumptions + handshake/profile notes.
Android key custody / exportability	Device identity private key is generated in the Android KeyStore under alias <code>dcs_device_identity</code> and is non-exportable; the internet-messaging identity key uses a KeyStore agreement-only key on API 31+ (in-keystore ECDH) with software fallback below; local symmetric key state and the SQLCipher seed are stored via EncryptedSharedPreferences.	Implementation key-custody note + Section 8 + privacy deployment note (<code>allowBackup=false</code>).
iOS key custody / exportability	Internet-messaging identity key prefers the Secure Enclave (in-enclave ECDH); Signal identity and session stores live in the Keychain with device-only protection classes; role-certificate device keys are Secure Enclave-backed P-256 with App Attest binding.	iOS conformance notes + Section 8.
Production key-recovery policy	Out-of-band QR reprovisioning or auditable admin provisioning only; handshake key-carry fields are non-authoritative/disabled in production trust decisions.	Protocol hardening note + risk R-06.

Field	Declared value	Evidence / provenance
Message-path boundary	Plaintext emergency content exists only on end-user devices. The offline path is local-radio device-to-device (BLE / RFCOMM default, role-gated Wi-Fi Aware extension, opt-in Public GATT mesh channel) with no cloud dependency; the internet path relays E2E-encrypted envelopes with delete-on-acknowledgement and TTL purge.	Cloud services section + Section 8 + Figure 3.
Source provenance anchor	Public source repository; current Android line 1.1.7/57.	Public source repository (GitHub) + retained March 31, 2026 source/build snapshots (tmp/evidence_20260331/*) for the 1.0.0 review pass.

A.4 Board one-page Go/No-Go card

This appendix card is intentionally page-isolated for decision-board packets.

Checklist block	GO condition	NO-GO trigger / required closure
Pilot-required artifacts	Build provenance, device matrix, signed responder workflow checklist, claim-evidence matrix, crypto profile sheet, risk-owner mapping, and (when hybrid features are in scope) the internet-plane assurance set are complete and reproducible.	Any artifact missing, unsigned, or not reproducible from declared paths/checksums.
Acceptance thresholds	Delivery, latency, voice envelope, power impact, operational readiness, and public-channel abuse-containment thresholds are all met; when hybrid features are in pilot scope, internet-path readiness thresholds (relay delivery, push wake-up, transport-handover continuity, call setup) are also met.	Any threshold fails without approved exception and dated mitigation plan.
Critical risks + mitigations	Risks R-01, R-02, R-05, R-06, R-07, and R-09 (data-protection posture of default-on registration) are closed with evidence-backed controls. R-03 (voice adaptive switching), R-04 (pilot log metadata), R-08 (internet-path expectation management), and R-10 (third-party provider dependency) are soft gates that MAY be closed during phased rollout provided the board explicitly records residual risk.	Mitigation remains narrative/process-only or lacks owner and timeline.
Public-mesh declared parameters	Hop/timestamp/rate/size/queue bounds and freshness labeling policy are frozen and auditable in logs/config snapshots.	Runtime drift from declared parameters or missing evidence for enforcement/downgrade behavior.
Trusted announcements	Responder-only announcement profile validation (signature, role scope, expiry) passes positive and negative tests; unverified payloads downgrade to untrusted rendering.	Any unverified announcement appears as trusted/high-priority alert in UI flow.
Out-of-scope acknowledgement	Board packet explicitly accepts non-goals (no nationwide anti-jamming guarantee, no full device-compromise defense, no unbounded unmanaged relay/store-forward).	Approval sought while assuming guarantees outside declared scope/non-goals.

A.5 T-08 closure protocol and command set

This subsection standardizes how reviewers can close pending abuse-governance evidence for claim C-06 / traceability T-08. For clarity, C-06/T-08 should be interpreted as runtime behavior closure work (stress counters + trusted-render downgrade proofs), not as already-complete unit-test coverage.

Primary artifacts.

- Protocol document: tmp/evidence_20260303/t08_closure_protocol.md
- Command-set script: tmp/evidence_20260303/collect_t08_evidence.sh

Command sequence (operator runbook).

```
EVID_DIR="./tmp/evidence_20260303"
"$EVID_DIR"/collect_t08_evidence.sh start RUN123
# Execute PM-01..PM-05 and TA-01..TA-06 scenarios
"$EVID_DIR"/collect_t08_evidence.sh stop RUN123
"$EVID_DIR"/collect_t08_evidence.sh summarize RUN123
```

Closure criterion. C-06/T-08 moves from Pending to Ready only when scenario logs confirm abuse-drop enforcement and zero trusted-render events for invalid/unverified announcement payloads.

A.6 Traceability matrix (reviewer-ready sample)

Claim ID	Doc section	Build	Evidence artifact	Status
T-01	7.1–7.12, 8, 13	v1.1.7 / 2.0.0	E-01, E-02, E-03, E-12.	Ready
T-02	10.2–10.3	v1.1.7 / 2.0.0	E-04, E-05, E-14.	Ready
T-03	9.6, A.3	CC-CRYPTO-PILOT-2.0	E-06, E-07.	Review
T-04	Evaluation Plan validation snapshot	Local QA snapshot 2026-03-31	E-08, E-16.	Review
T-05	A.2	v1.1.7 access flow	E-09, E-13.	Controlled
T-06	A.2, A.3	Public source repository + retained March 31, 2026 build-config snapshot (1.0.0 review pass; refreshed v1.1.7 snapshot pending)	E-10.	Review
T-07	A.2	internal repo	E-11, E-15.	Controlled
T-08	Public GATT profile, Evaluation Plan abuse controls, A.4–A.5	v1.1.7 / 2.0.0	E-17, E-18, E-19, E-20.	Pending
T-09	Section 8, internet-path crypto profiles, A.3	v1.1.7 + iOS 1.1.2 + web / 2.0.0	E-48, E-49; field closure tracked as G-06..G-09.	Review

Evidence link index (for Traceability matrix).

ID	Evidence artifact / link
E-01	Figure 1 (RFCOMM lifecycle).
E-02	Figure 3 (message-path cloud separation).
E-03	Technical walkthrough page: https://crisisconnect.network/technical .

ID	Evidence artifact / link
E-04	Rescue role-proof validation rules.
E-05	Figure 2 (Rescue Mode promotion gates).
E-06	Cryptographic profile lock controls.
E-07	Appendix A.3 (CC-CRYPTO-PIL0T-2.0 profile sheet).
E-08	Current validation snapshot and local QA handoff notes for the Android v1.1.7 review pass.
E-09	Reviewer access and release coordination references captured in Section A.2.
E-10	Source provenance link: https://github.com/emirhan-duman/Crisis-Connect plus March 31, 2026 source/build snapshot artifacts.
E-11	Owner-controlled internal Android repository and local workspace evidence documented in Section A.2 (access-controlled reviewer workflow).
E-12	Local source-provenance artifact bundle: tmp/evidence_20260331/source_provenance.txt.
E-13	Local build-config artifact bundle: tmp/evidence_20260331/build_config_snapshot.txt.
E-14	Local Public-mesh gating artifact bundle: tmp/evidence_20260303/public_mesh_gating.txt.
E-15	Local Public-GATT packet-format artifact bundle: tmp/evidence_20260303/public_gatt_packet_format.txt.
E-16	Local unit-test artifact bundle: tmp/evidence_20260331/unit_test_summary.txt; release QA may attach a companion HTML or CI export when generated.
E-17	Pilot-run Public-mesh abuse counter export (invalid timestamp, hop-limit stop, inbound-rate drops, oversize drops, duplicate drops): tmp/evidence_20260303/public_mesh_abuse_counters.json.
E-18	Trusted-announcement validation report (valid signature, expired payload, wrong role, bad signature, downgrade-to-untrusted UI evidence): tmp/evidence_20260303/trusted_announcement_validation.md.
E-19	T-08 closure protocol (scenario matrix + closure gates): tmp/evidence_20260303/t08_closure_protocol.md.
E-20	T-08 command-set evidence collector (start/stop/summarize): tmp/evidence_20260303/collect_t08_evidence.sh.
E-21	Pixel 7a paired standby power comparison summary (March 15, 2026): tmp/evidence_20260315/standby_power_trace_summary.txt.
E-22	Pixel 7a baseline idle raw Perfetto trace (March 15, 2026): tmp/evidence_20260315/pixel7a_idle_baseline.trace.
E-23	Pixel 7a app-standby raw Perfetto trace (March 15, 2026): tmp/evidence_20260315/pixel7a_app_standby.trace.

ID	Evidence artifact / link
E-24	Adjacent-device voice baseline summary (March 15, 2026): tmp/evidence_20260315/voice_gate_run_01/voice_call_baseline_summary.txt.
E-25	Samsung SM-J260F voice baseline raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_gate_run_01/sm_j260f_rfcomm_dump.log.
E-26	Samsung SM-G990E voice baseline raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_gate_run_01/sm_g990e_rfcomm_dump.log.
E-27	Two-wall voice continuity summary (March 15, 2026): tmp/evidence_20260315/voice_gate_run_02/voice_call_two_walls_summary.txt.
E-28	Samsung SM-J260F two-wall voice raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_gate_run_02/sm_j260f_rfcomm_run2.log.
E-29	Samsung SM-G990E two-wall voice raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_gate_run_02/sm_g990e_rfcomm_run2.log.
E-30	Voice latency instrumentation summary (March 15, 2026): tmp/evidence_20260315/voice_latency_run_02/voice_latency_instrumentation_summary.txt.
E-31	Samsung SM-J260F voice latency raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_latency_run_02/sm_j260f_rfcomm_latency.log.
E-32	Samsung SM-G990E voice latency raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_latency_run_02/sm_g990e_rfcomm_latency.log.
E-33	Voice clock-adjusted latency summary (March 15, 2026): tmp/evidence_20260315/voice_latency_run_05/voice_clock_adjusted_latency_summary.txt.
E-34	Samsung SM-J260F clock-adjusted voice latency raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_latency_run_05/sm_j260f_rfcomm_latency.log.
E-35	Samsung SM-G990E clock-adjusted voice latency raw RfcommService dump (March 15, 2026): tmp/evidence_20260315/voice_latency_run_05/sm_g990e_rfcomm_latency.log.
E-36	iOS deployment-baseline kit overview and operator workflow: tmp/evidence_20260403_ios_baseline/README.md.
E-37	iOS release-manifest template for 1.1.2 promotion work: tmp/evidence_20260403_ios_baseline/release_manifest_template.md.
E-38	iOS assurance-report template for reviewer handoff: tmp/evidence_20260403_ios_baseline/assurance_report_template.md.

ID	Evidence artifact / link
E-39	iOS device-matrix template for Apple hardware coverage: tmp/evidence_20260403_ios_baseline/device_matrix_template.csv.
E-40	iOS scenario-matrix template for repeatable run capture: tmp/evidence_20260403_ios_baseline/scenario_matrix_template.md.
E-41	iOS claim-to-evidence and limitations templates: tmp/evidence_20260403_ios_baseline/claim_evidence_map_template.md and tmp/evidence_20260403_ios_baseline/known_limitations_template.md.
E-42	Current signed iOS archive snapshot for the April 3, 2026 candidate: tmp/evidence_20260403_ios_baseline/signed_archive_snapshot_2026-04-03.md.
E-43	iOS field-validation playbook for structured device-run closure: tmp/evidence_20260403_ios_baseline/field_validation_playbook_2026-04-04.md.
E-44	iOS session execution checklist for live validation runs: tmp/evidence_20260403_ios_baseline/session_execution_checklist_2026-04-04.md.
E-45	First physical iPad deployment snapshot for the current iOS candidate: tmp/evidence_20260403_ios_baseline/physical_ipad_deployment_snapshot_2026-04-04.md.
E-46	First manual iPad workflow snapshot for the current iOS candidate: tmp/evidence_20260403_ios_baseline/physical_ipad_manual_workflow_snapshot_2026-04-05.md.
E-47	First basic iPad-to-Android interoperability snapshot for the current iOS candidate: tmp/evidence_20260403_ios_baseline/ios_android_basic_interop_snapshot_2026-04-05.md.
E-48	Internet-plane source-review basis: hybrid transport, envelope v2/v3, SPAKE2, call, and MLS implementations with cross-platform golden-vector and unit test suites in the Android and iOS client repositories (July 2026 line).
E-49	Web control-plane July 2026 expansion review basis: messaging/SFU implementation and security rules, assistant-channel routes and secret handling, workflow scheduler/cron/queue and MCP routes, Playwright end-to-end suites, and deployment (Compose/Helm) additions in the crisis-connect-web workspace.

A.7 Disclaimer & Limitation of Liability

This document and the associated software artifacts are provided for pilot evaluation and institutional review purposes only. They are not a certification record, legal opinion, or a guarantee of field performance in all environments.

- **No replacement for public emergency systems.** Crisis Connect does not replace national emergency numbers (for example 112/911) or certified public-safety infrastructure. Users and operators **MUST** attempt official emergency channels whenever available.
- **No delivery or availability guarantee.** Message and voice delivery can fail due to radio conditions, device constraints, OS background policy, power state, interference, or implementation defects.
- **As-is / as-available basis.** To the maximum extent permitted by applicable law, the system and documentation are provided without express or implied warranties, including fitness for a particular purpose, merchantability, and non-infringement.
- **Operational responsibility remains with deployers.** Institutions deploying the system are responsible for training, fallback communication procedures, incident command policy, and local validation before operational use.
- **Limitation of liability.** To the maximum extent permitted by law, Auralis Industries and contributors are not liable for indirect, incidental, special, consequential, or punitive damages, or for loss of data, revenue, or operational opportunity arising from use or inability to use the system.
- **Jurisdictional compliance.** Deployer organizations are responsible for compliance with local telecom, emergency-communications, privacy, and consumer-protection requirements. If local law restricts any disclaimer or liability limitation in this subsection, that restriction applies only to the required extent.

© 2026 Auralis Industries.