Services & Listeners

A map of what CrossConnect runs while it is up: every listening socket, every scheduled job, every inbound endpoint, and whether each is on or off out of the box. A platform or security engineer should be able to build firewall rules and an operations runbook straight from this page.

Audience: platform / infrastructure engineering, network security, operations

Scope: network listeners, scheduled jobs, the service-bean surface, inbound REST receivers, outbound dispatchers, bind ports & enable flags

Posture: all five listeners off by default · no socket opens until you enable it · data leaves the deployment only where you configure a destination

Document: services & listeners reference, 21 June 2026

Contact: contact_us@cybriq.io

0 How to read this document

This reference names concrete mechanisms, not adjectives. For each item it gives the class that opens the socket, the port it uses, the setting that controls it, and the default on/off value that decides whether the socket opens at all. Anything that is an optional add-on rather than a default is labelled as such. You should be able to write a host firewall policy, an outbound allow-list, and a pause/purge runbook from this page alone, without reading the source.

GA bound and running with the default config

CONFIGURABLE built in, but an operator has to switch it on (at runtime or via a property)

DEPLOYMENT OPTION off by default; opens no socket until you enable it

1 Runtime surface at a glance
2 The service-bean surface
3 Network listeners & bind ports
4 How a listener is gated & toggled
5 Scheduled jobs & cadence
6 Inbound HTTP receivers
7 Outbound dispatchers & egress
8 Firewall & runbook cheat sheet
A Listener bind matrix
B Scheduled-job matrix
C Enable-flag reference

1 Runtime surface at a glance

CrossConnect is a single Spring Boot application (Java 21, Spring Boot 3.4.0) backed by PostgreSQL, with an optional formal-analysis sidecar. While it runs, it has four kinds of moving part: a thin HTTP/UI layer that hands work to **services**; a set of passive **network listeners** that take in data the network offers; a group of **scheduled jobs** on timers; and a few **outbound dispatchers** that send events out. Four facts frame every control on this page.

One inbound app port by default

On a fresh install the only socket listening for connections is the application port (`server.port`, default `8080`). All five network listeners stay off and open no socket until you turn them on.

Listeners are passive & signal-only

The flow, trap, mDNS, and DHCP receivers take in only what the network already broadcasts. They read protocol metadata and queue an observation. None reads packet contents or runs SPAN/mirror capture.

Jobs split collect vs. housekeep

The heavy collection jobs (discovery, reachability) are controlled by a database flag and stay off by default. The housekeeping jobs that trim tables (staging, audit, webhook, soft-delete) are always on, so no table can grow forever. Automation is opt-in.

Egress only where configured

Outbound webhooks, SIEM/chat sinks, the identity provider (IdP), and the optional language model are contacted only when you give them a destination. Leave the URL blank and that path is off. Nothing leaves the deployment behind your back.

> **The off-by-default rule.** The flow, SNMP-trap, mDNS, and DHCP-fingerprint listeners all open **no socket** on a default build. Two of them are controlled by a Spring `@ConditionalOnProperty`, which means the bean is not even created until you flip the property. The other two do load their bean, but keep the socket closed until an operator enables them at runtime. Either way, a default deployment exposes the application port and nothing else.

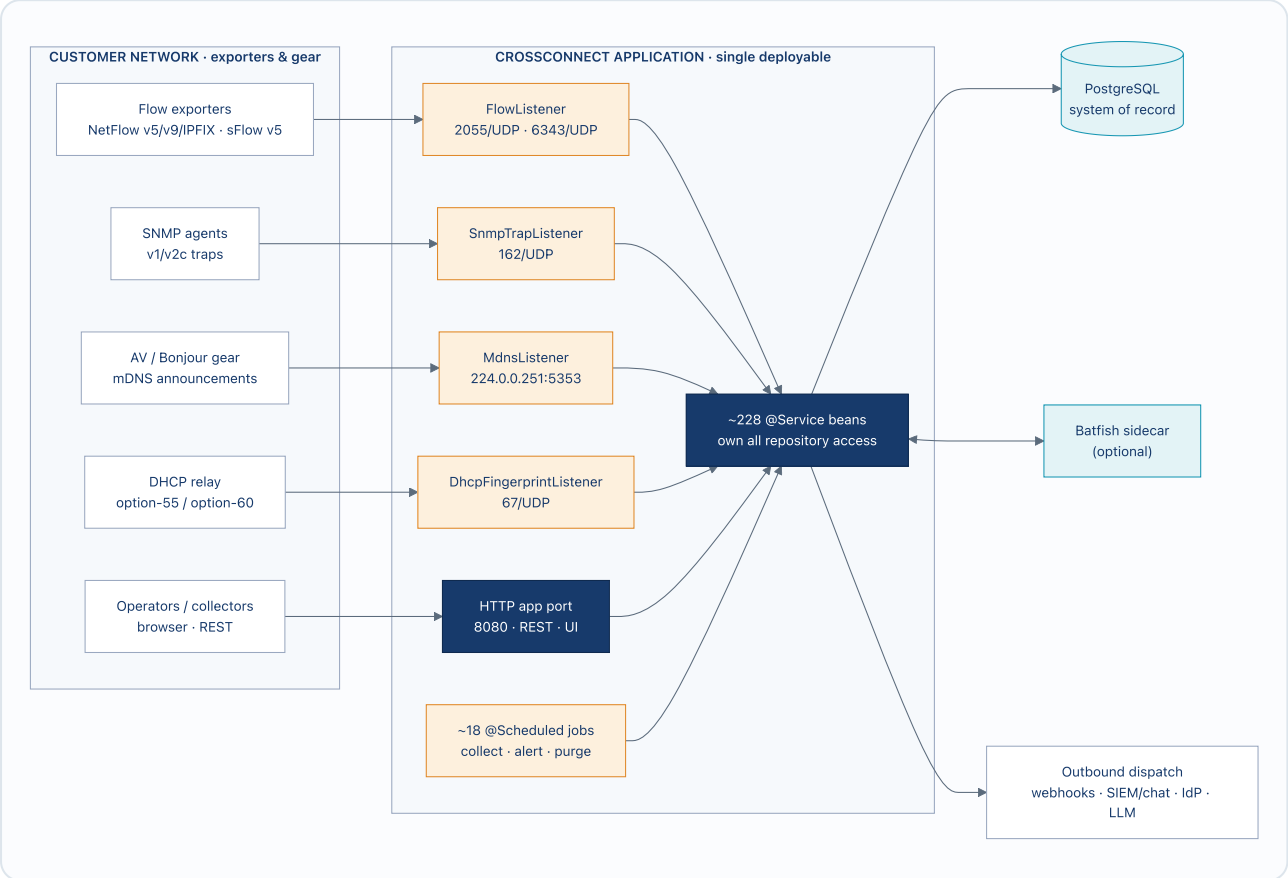

Figure 1. What runs while CrossConnect is up. Five passive listeners (amber, all off by default) and the HTTP app port feed the service layer, which owns all data access and drives the scheduled jobs and outbound dispatchers. On a default build only the app port opens; every listener stays off until you enable it.

2 The service-bean surface

The application is built in layers. The HTTP controllers (around 115 `@RestController` classes) and the Vaadin UI views stay thin and hand their work to **services**, and the services are the only code that touches the database. The codebase carries roughly **228** `@Service` beans. That count is high on purpose: each service handles one concern, which keeps every piece small and easy to test. Internal collectors, schedulers, and shared plumbing are services too, but they are not user features. A check in the build enforces that every user-facing capability shows up on the UI, the catalog, the glossary, and the AI surfaces; genuinely internal beans simply omit the marker that triggers that check.

Domain	~Count	What this group does · representative beans
Discovery	19	Find and stage what is on the network. `DiscoveryWorker, DiscoveryRunner, SnmpModelInference, RogueDeviceService, CollectorSettingsService, ManagementRangeService, RangeSweepService`
Inventory & config	30	The canonical CRUD model. `DeviceService, CableService, RackService, ModuleService, LocationService, CircuitService, TrafficFlowService, ConfigContextService`
IPAM	6	Address management. `PrefixService, IpAddressService, IpRangeService, AggregateService, IpAddressSpaceService, IpamRoleService`
Topology & L2/L3	13	Ports, VLANs, VRFs, services. `InterfaceService, VlanService, VrfService, OuiResolver, NetworkServiceService, ServiceSuggestionService`
AV / media	12	Audio-visual inventory and timing. `AvClassificationService, AvEndpointClassifier, AvMediaFlowService, PtpService, MulticastGroupService, ApRadioService`
Multicast & networking	13	Multicast health and overlays. `MulticastConfigAnalyzer, MulticastHealthService, MulticastTroubleshooter, RpAnalyzer, MulticastAlertService, TunnelService`
Batfish & analysis	10	Formal config reasoning. `BatfishSnapshotService, ReachabilityService, AclAnalysisService, ForwardingHealthService, FailureImpactService, ChangePreviewService`
Assurance & operations	16	Risk, capacity, change safety. `HotspotService, CapacityPlanningService, BlackBoxService, BlastRadiusService, ThreatDetectionService, GoldenConfigService`
Hardening & security	5	Config hardening and CVEs. `ConfigHardeningService, ConfigHardeningAnalyzer, ConfigGradeService, CveService, NvdLookupService`
Compliance	4	Frameworks as data. `ComplianceService, CustomControlService, ComplianceGapService, EvidencePackService`
AI & conversational	4	The grounded assistant. `AssistantService, WriteIntentService, AiQualityService, AiSettingsService`
Lifecycle & zones	4	EoL and trust planes. `LifecycleService, EolLookupService, NetworkZoneService, SegmentationService`
Integrations & automation	9	Inbound trust, event rules, sinks. `InboundValidationService, OutboundSinkService, ConnectorRegistry, EventRuleService, AutomationWorker, ApiTokenService`

Domain	~Count	What this group does · representative beans
Admin, auth & reporting	12	Tenancy, identity, reports. `UserService, PermissionService, JwtTokenService, OidcService, ReportService, ScheduledReportSweep, UpgradeService`
Occupancy & wireless	8	Wi-Fi analytics & presence. `OccupancySnapshotSweep, OccupancyRetentionSweep, OccupancyFileSink, WifiCoverageService, WirelessService`
Cross-cutting & misc	29	Audit, search, atlas, data quality. `EventAuditService, ActivityFeedService, NetworkAtlasService, DataQualityService, MaturityService, SearchService, DemoSeedService`

> **Why there are so many services.** Each noun gets its own service (`DeviceService`), each analysis gets one (`FailureImpactService`), and each cross-cutting capability gets one (`EventAuditService`). None of these beans opens a network socket. The listeners and schedulers in §3 and §5 are the only components that open sockets or run on timers. The counts here are approximate and shift as the build evolves.

3 Network listeners & bind ports

Five passive receivers can open a UDP socket or join a multicast group to take in data the network already sends. Each one is **off by default**: a fresh install opens none of these ports. When enabled, a receiver reads only the protocol metadata, matches the sender to a known device, and queues an observation through a service. It never reads packet contents and never captures traffic. The two flow receivers (NetFlow and sFlow) share one bean (`FlowListener`) but use two sockets.

Listener (class)	Binds	Socket	Captures / used for	Default
mDNS `MdnsListener`	`224.0.0.251:5353/UDP` (multicast)	`MulticastSocket`, joins group	The Bonjour/mDNS service announcements that AV gear broadcasts (`_airplay._tcp` , `_qsys._tcp` and similar). This is the best clue for identifying AV endpoints short of capturing traffic; it queues a `DiscoveredMdnsService` .	OFF
DHCP fingerprint `DhcpFingerprintListener`	`67/UDP` (relayed)	`DatagramSocket`	A relayed copy of DHCP DISCOVER/REQUEST messages, forwarded by the switch `ip helper-address` . It reads the option-55 / option-60 fingerprint to recognize a device family, then queues a `DiscoveredDhcpFingerprint` .	OFF

Listener (class)	Binds	Socket	Captures / used for	Default
SNMP traps `SnmpTrapListener`	`0.0.0.0:162/UDP` (v1/v2c)	snmp4j `DefaultUdpTransportMapping`	Real-time device notifications (link down, cold start, PSU fail, signal loss). Each one is sorted into an observation kind and handed to `InboundValidationService`, which emits a `RecordChangeEvent` to the activity feed and audit.	OFF
NetFlow `FlowListener`	`2055/UDP` (configurable)	`DatagramSocket`	NetFlow v5/v9/IPFIX exported straight from routers and switches. `NetflowDecoder` turns it into conversations and top-talkers, which `TrafficFlowService` writes. No packet contents.	OFF
sFlow `FlowListener`	`6343/UDP` (configurable)	`DatagramSocket`	sFlow v5 samples, decoded by `SflowDecoder`. These are stored and used the same way as NetFlow (the `POST /api/v1/flows` push API also feeds the same `TrafficFlowService`).	OFF

> **What no listener does:** none of them reads packet contents, runs deep packet inspection, or consumes a SPAN/mirror feed. They read protocol metadata only (flow summaries, trap varbinds, service-announcement names, DHCP option fingerprints) and queue it as an untrusted observation for the trust gate to score. The active probes in the discovery layer (mDNS/SSDP/WS-Discovery, which query and collect) are **not** listeners and open no listening socket.

Besides these sockets, the flow path can also accept an HTTP push instead of the UDP receivers, and external systems can submit claims over HTTP to the inbound trust gate. Both are covered in §6.

4 How a listener is gated & toggled

A listener is enabled in one of two ways, depending on how it is meant to be operated. Infrastructure-level receivers are set once at deploy time with a Spring property. Segment-level receivers are switched on per segment from the UI while the app runs. Either way, no socket opens until the listener is enabled.

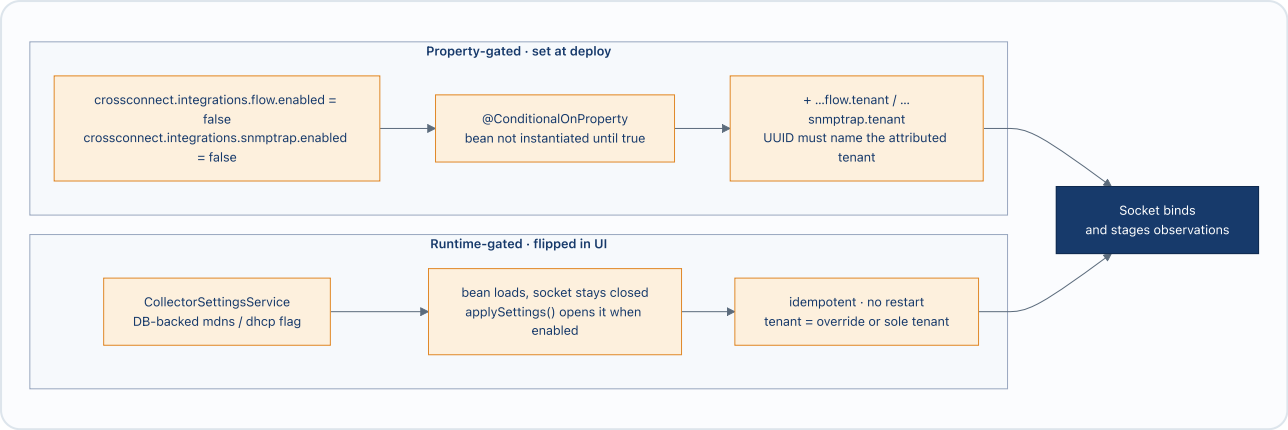

Figure 2. Two ways to enable a listener. NetFlow/sFlow and SNMP traps are controlled by `@ConditionalOnProperty` plus a tenant UUID, so on a default build the bean is not even created. mDNS and DHCP do load their bean but keep the socket closed; an operator flips the DB-backed `CollectorSettingsService` flag in the UI, and an idempotent `applySettings()` opens the socket with no restart.

Property-gated (flow, SNMP traps)

The bean is annotated `@ConditionalOnProperty(...="…enabled", havingValue="true")`, so it does not exist until you set that property. Even then it does nothing until a matching `…tenant` UUID names the tenant the data belongs to. A fresh install opens no flow or trap port.

DEPLOYMENT OPTION

Runtime-gated (mDNS, DHCP)

The bean always loads but stays idle until the DB-backed `CollectorSettingsService` reports it as enabled. An operator turns it on per segment from the UI with one click and no restart. The tenant is an optional override (`…mdns.tenant` / `…dhcp.tenant`); if you leave it unset on a single-collector install, it resolves to the one tenant present.

RUNTIME

5 Scheduled jobs & cadence

Scheduling is handled by `SchedulingConfig` (`@EnableScheduling`, controlled by `crossconnect.scheduling.enabled`, on by default; tests turn it off). Eighteen jobs keep the data current and the tables tidy. The cadences below are the resolved defaults of `${property:default}` placeholders, and every interval is tunable. Collection jobs are controlled by a database flag and stay off by default; the housekeeping purges are always on; automation and the BI export are opt-in.

Job (class · method)	Cadence (default)	What it does	State
`DiscoveryWorker.sweep`	fixedRate 5 min (delay 1 min)	Sweeps every active tenant/device (SNMP/LLDP) and stages results.	OFF BY DEFAULT
`ReachabilityCollector.collect`	fixedRate 2 min (delay 1.5 min)	ICMP/TCP probes per managed IP; records reachability for health rollups.	OFF BY DEFAULT
`DriftSweep.sweep`	fixedRate 15 min	Detects golden-config drift; emits a drift event to audit + webhooks.	ALWAYS ON
`MulticastAlertService.scan`	fixedRate 5 min	Diffs multicast state (groups/sources/queriers); fires change events.	ON BY DEFAULT

Job (class · method)	Cadence (default)	What it does	State
`AvReadinessDriftService.scan`	fixedRate 1 h	Fires when AV readiness regresses past an 8-point threshold from baseline.	ON BY DEFAULT
`BatfishHealth.scheduledProbe`	fixedDelay 1 min	Liveness probe of the Batfish sidecar; surfaces a UI banner if unreachable.	ON WHEN URL SET
`BatfishWarmer.rewarmPeriodically`	fixedDelay 5 min	Re-warms per-tenant snapshots so analyses stay hot; content-addressed cache.	ON BY DEFAULT
`WriteIntentSweep.sweep`	fixedRate 1 min	Expires PROPOSED AI write-intents past their 15-min TTL (confirm-before-commit).	ALWAYS ON
`ScheduledReportSweep.sweep`	fixedRate 1 min	Delivers due report subscriptions by email.	ALWAYS ON
`UpgradeNotificationSweep.sweep`	fixedRate 5 min	Fires due upgrade advance-notifications.	ALWAYS ON
`AutomationWorker.evaluateAll`	fixedRate 10 min	Evaluates per-tenant event rules; can fire webhooks/jobs.	OFF BY DEFAULT
`OccupancySnapshotSweep.capture`	fixedRate 15 min	Captures a per-tenant Wi-Fi occupancy snapshot (dormant with no AP source).	ALWAYS ON
`OccupancyRetentionSweep.sweep`	fixedRate 1 h	Rolls up and prunes occupancy samples so the series stays bounded.	ALWAYS ON
`OccupancyFileSink.drop`	fixedRate 15 min	Drops a per-tenant CSV export for BI ingestion; idle until export-dir set.	OFF BY DEFAULT
`StagingPurgeSweep.sweep`	fixedRate 24 h (delay 1 h)	Prunes 17 `discovered_*` staging tables past the retention window.	ALWAYS ON
`EventAuditPurgeSweep.sweep`	fixedRate 24 h (delay 1 h)	Drops `event_audit` rows past retention.	ALWAYS ON
`WebhookDeliveryPurgeSweep.sweep`	fixedRate 24 h (delay 1 h)	Drops `webhook_delivery` records past retention.	ALWAYS ON
`DeviceSoftDeletePurgeSweep.sweep`	fixedRate 24 h (delay 1 h)	Hard-deletes soft-deleted devices past the grace window (set ≤0 to disable).	ALWAYS ON

Design split. The expensive collection jobs (discovery, reachability) are controlled by the DB-backed `DiscoverySettings.enabled` flag, so operators can quiet the network from the UI without a restart. They are off on a fresh install. The housekeeping purges are always on, so no table grows without bound. Automation (`crossconnect.automation.enabled`) and the BI export file sink stay off until you opt in. Default retention windows are listed in §8 and Appendix B.

6 Inbound HTTP receivers

All HTTP traffic arrives on the single application port (default `8080`, or `443` behind your TLS terminator). Beyond the ordinary create/read/update/delete endpoints, a handful of controllers under `/api/v1/*` act as inbound *receivers*: external systems push facts in, but those facts are never silently accepted as truth. Several of these stay off until a property enables them, and the occupancy webhooks require an issued API key rather than just a tenant header.

Endpoint	Controller	Purpose & auth	State
`POST /api/v1/flows`	`TrafficFlowController`	A collector pushes flow conversation summaries here, as an alternative to the UDP receivers; both land in the same `TrafficFlowService` store. Tenant is set via `X-CrossConnect-Tenant`.	GA
`POST /api/v1/inbound/event`	`InboundEventController`	External systems POST claimed facts to the trust gate. Each one is queued as a review proposal and never applied automatically. Loaded only when `crossconnect.integrations.inbound.enabled=true`.	AUTHN-GATED
`POST /api/v1/discovery/observations`	`DiscoveryIngestController`	A per-segment collector posts parsed observations here (mDNS/SSDP/ARP). Uses `X-CrossConnect-Tenant` plus an optional `X-CrossConnect-Collector-Token`.	TOKEN-GATED
`POST /api/v1/occupancy/mist/zone-events`	`MistZoneWebhookController`	Turns wireless-vendor zone enter/exit events into anonymized dwell visits; the client handle is hashed and then discarded. Requires an issued `X-API-Key`.	KEY-GATED
`POST /api/v1/occupancy/cisco-spaces/events`	`CiscoSpacesWebhookController`	Turns a wireless-vendor occupancy event stream (`DEVICE_COUNT`, `SPACE_OCCUPANCY`) into per-space samples. Requires an issued `X-API-Key`.	KEY-GATED
`GET /api/v1/metrics`	`MetricsController`	A Prometheus-format scrape endpoint for your existing dashboards. Loaded only when `crossconnect.integrations.metrics.enabled=true`.	OFF BY DEFAULT

> **Nothing is accepted on faith.** Every inbound claim (flows, discovery observations, asserted events, occupancy) lands as an untrusted observation. It is checked against the source of truth, scored, and audited before anything happens, and it is never written straight into the documented model. The application also exposes Spring Actuator health/readiness and (optionally) Prometheus on the same app port. The OTLP tracing exporter is an *outbound* client (default endpoint `localhost:4318`, export off at runtime until you toggle it), not a listener.

7 Outbound dispatchers & egress

Three components carry events out of the system. All three are driven by the internal event bus, and all are safe by default: with no destination configured, nothing leaves. These three are the entire outbound surface you need to allow-list.

Dispatcher (class)	What it does	Egress
`EventBus`	The in-process publish/subscribe backbone. Every record change is published as a `RecordChangeEvent`, and audit, webhooks, and sinks subscribe to it. Dispatch is synchronous; a failing subscriber is logged and does not take the rest down. Nothing leaves the process without passing through here first.	none (internal)
`WebhookDispatcher`	Matches each event to the per-tenant subscriptions, signs the body with HMAC-SHA256 (header `X-CrossConnect-Signature`, hex), and POSTs it. On a 3xx/5xx/408/429 or network error it retries with growing back-off (1s · 5s · 25s · … capped at 1h, up to 6 attempts). It saves each `WebhookDelivery` and picks up pending retries again on startup.	HTTPS to your receiver
`OutboundSinkService`	A lightweight export rail: it ships `RecordChangeEvent`s to a SIEM HTTP collector and/or a chat webhook, set per tenant at runtime (env values act as the fallback default). Each is a single fire-and-forget POST with no retries; leave the URL blank and that sink is off.	HTTPS to SIEM / chat

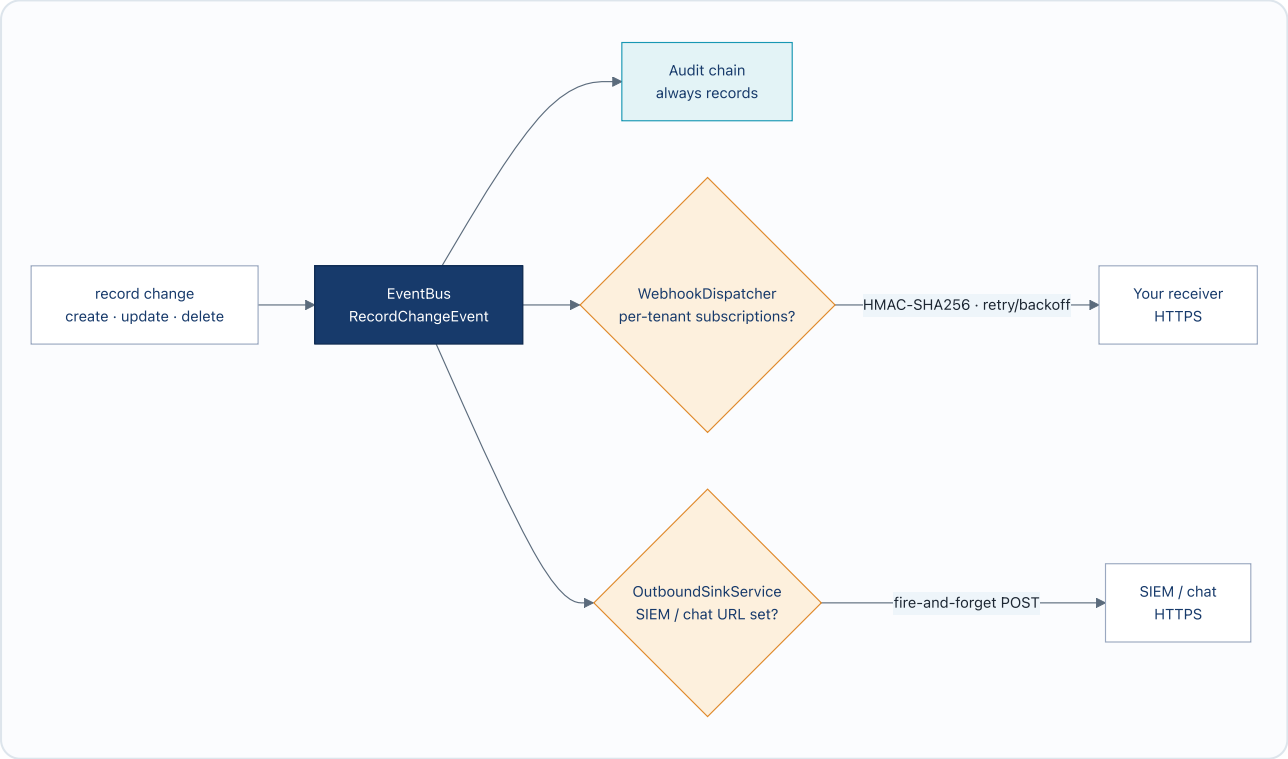

Figure 3. One event, three subscribers. The audit chain always records the change. The webhook rail delivers it, signed and reliably, wherever a subscription exists. The sink rail ships it fire-and-forget only where a SIEM or chat URL is set. With no destinations configured, the only effect is the audit write, and no packet leaves the deployment.

8 Firewall & runbook cheat sheet

The short version: everything an operator needs to write a host firewall policy and a pause/purge runbook.

Inbound to allow (default build)

Just `8080/TCP` (or your TLS port) to the app. Open a listener's UDP port *only after* you enable that listener: `5353` (mDNS, multicast), `67` (DHCP), `162` (SNMP traps), `2055` (NetFlow), `6343` (sFlow).

Outbound to allow

`161/UDP` + `22/TCP` + `443/TCP` to managed gear/cloud (discovery); `5432/TCP` to PostgreSQL; sidecar RPC to Batfish; `443` to your webhook/SIEM/chat, IdP, and optional LLM; `4318` to your OTLP collector if tracing export is on.

Quiet the network (no restart)

Turn off `DiscoverySettings.enabled` in the UI to pause `DiscoveryWorker` and `ReachabilityCollector`. Flip the mDNS/DHCP collector flags off to close those sockets. The property-gated flow and trap listeners need a config change and a restart.

Retention windows (defaults)

Staging `discovered_*` 14 days; audit 90 days; webhook delivery 30 days; device soft-delete grace 90 days; AI write-intent TTL 15 min. Each is tunable; the purge sweeps run every 24 h and are always on.

Default-build summary. One inbound socket (the app port). Zero listener sockets. Outbound traffic only where you configure a destination. Collection stays paused until you enable it; housekeeping is always on. From this page you can scope a least-exposure firewall policy and a pause/purge runbook without reading the source.

A Appendix, listener bind matrix

Listener	Bind	Enable flag	Default	Tenant gate	Toggle
mDNS	224.0.0.251:5353/UDP	`crossconnect.discovery.mdns.enabled` (+ `CollectorSettingsService`)	`false`	… `mdns.tenant` or sole tenant	UI runtime (DB-backed)
DHCP fingerprint	67/UDP	`crossconnect.discovery.dhcp.enabled` (+ `CollectorSettingsService`)	`false`	… `dhcp.tenant` or sole tenant	UI runtime (DB-backed)
SNMP traps	0.0.0.0:162/UDP	`crossconnect.integrations.snmptrap.enabled`	`false`	… `snmptrap.tenant` (required)	property + restart

Listener	Bind	Enable flag	Default	Tenant gate	Toggle
NetFlow	2055/UDP	`crossconnect.integrations.flow.enabled`	`false`	… `flow.tenant` (required)	property + restart
sFlow	6343/UDP	`crossconnect.integrations.flow.enabled`	`false`	… `flow.tenant` (required)	property + restart
App HTTP port	8080/TCP (or 443)	`server.port`	`8080` (bound)	per-request tenant filter	always (the only default socket)

B Appendix, scheduled-job matrix

Job	Type	Default cadence	Gate	Category
DiscoveryWorker	fixedRate	5 min	DiscoverySettings.enabled (off)	discovery
ReachabilityCollector	fixedRate	2 min	DiscoverySettings.enabled (off)	reachability
DriftSweep	fixedRate	15 min	always on	drift
MulticastAlertService	fixedRate	5 min	multicast.alerts.enabled (on)	alert
AvReadinessDriftService	fixedRate	1 h	av.drift.enabled (on)	drift
BatfishHealth	fixedDelay	1 min	batfish.url set	health
BatfishWarmer	fixedDelay	5 min	warmers.enabled (on)	warm
WriteIntentSweep	fixedRate	1 min	always on	intent expiry
ScheduledReportSweep	fixedRate	1 min	always on	report
UpgradeNotificationSweep	fixedRate	5 min	always on	notification
AutomationWorker	fixedRate	10 min	automation.enabled (off)	automation
OccupancySnapshotSweep	fixedRate	15 min	always on (dormant w/o source)	collector
OccupancyRetentionSweep	fixedRate	1 h	always on	retention
OccupancyFileSink	fixedRate	15 min	export-dir set (off)	BI export
StagingPurgeSweep	fixedRate	24 h	always on (14-day retention)	purge
EventAuditPurgeSweep	fixedRate	24 h	always on (90-day retention)	purge

Job	Type	Default cadence	Gate	Category
WebhookDeliveryPurgeSweep	fixedRate	24 h	always on (30-day retention)	purge
DeviceSoftDeletePurgeSweep	fixedRate	24 h	≤0 disables (90-day grace)	purge

C Appendix, enable-flag reference

The main properties that decide whether a listener or job is live. The defaults aim for a quiet, safe out-of-the-box posture. Ports and intervals are tunable through the matching `…port` / `…interval-ms` keys.

Surface	Property	Default
App HTTP port	`server.port` (`SERVER_PORT`)	`8080`
Scheduling master	`crossconnect.scheduling.enabled`	`true`
Discovery sweep	`crossconnect.discovery.enabled` (+ DB `DiscoverySettings`)	`false`
SNMP trap listener	`crossconnect.integrations.snmptrap.enabled` / `…port`	`false` / `162`
Flow listeners	`crossconnect.integrations.flow.enabled` / `…netflow-port` / `…sflow-port`	`false` / `2055` / `6343`
mDNS listener	`crossconnect.discovery.mdns.enabled` / `…port`	`false` / `5353`
DHCP listener	`crossconnect.discovery.dhcp.enabled` / `…port`	`false` / `67`
Inbound event API	`crossconnect.integrations.inbound.enabled`	`false`
Metrics endpoint	`crossconnect.integrations.metrics.enabled`	`false`
Automation worker	`crossconnect.automation.enabled`	`false`
Batfish sidecar	`crossconnect.batfish.url`	`http://localhost:8888`
SIEM / chat sinks	`crossconnect.integrations.siem.url` / `…chat.url`	empty (off)
Staging retention	`crossconnect.discovery.staging.retention-days`	`14`
Audit retention	`crossconnect.audit.retention-days`	`90`
Webhook retention	`crossconnect.webhooks.retention-days`	`30`
Soft-delete grace	`crossconnect.device.soft-delete-retention-days`	`90`
AI write-intent TTL	`crossconnect.ai.intent-ttl-seconds`	`900`

CrossConnect by CybrIQ · Services & Listeners Reference · Technical reference · 21 June 2026 · Listeners, ports, and defaults reflect the shipped operator-preview build; counts are approximate and items marked *deployment*

option bind no socket until enabled. · contact_us@cybriq.io