CrossConnect
CONNECT. MONITOR. OPTIMIZE.
BY Cybr IQ

Data Collection Reference

A plain account of what CrossConnect reads off each device, over which protocol, and how every value is staged and validated into the source of truth. Written for the network engineer who wants the OIDs, the ports, the libraries, and the staging table behind each fact, not adjectives.

Audience: network engineering, NetEng tooling, security review

Scope: every collector, MIB column, OID, listener, CLI command, and application input

Stack: Java 21 · Spring Boot 3.4.0 · PostgreSQL · snmp4j 3.8.2 · sshj 0.38.0 · Batfish

Document: technical reference, 21 June 2026

Contact: contact_us@cybriq.io

0 How to read this document

Every collector below is a real protocol implementation, not a mock-up. Each section names the mechanism rather than an adjective: the OID or column walked, the port bound, the library and pinned version, the staging entity written, and the canonical record it becomes. Collectors that are opt-in or off by default are labelled, so you can tell which sockets a stock deployment opens.

ON active by default in a discovery sweep **OPT-IN** shipped, operator-enabled by flag or credential

OFF dormant listener, opens no network socket until enabled

1 How collection works

CrossConnect reads, it does not intercept. Every input is either a control-plane query the network already answers, an announcement the gear already broadcasts, configuration text, or a record an operator or another system hands in. Nothing is sniffed off the wire: there is no packet capture, no SPAN or mirror feed, and no payload inspection anywhere in the platform.

Four families of collector feed one pipeline. Each writes to **staging**, an append-only set of `discovered_*` observations. Validation then confidence-scores each staged observation and commits it into the **source of truth**, the canonical store every read, view, and AI answer resolves against. The derived layer sits on top: it computes scores and rollups from that source of truth and writes no new facts of its own.

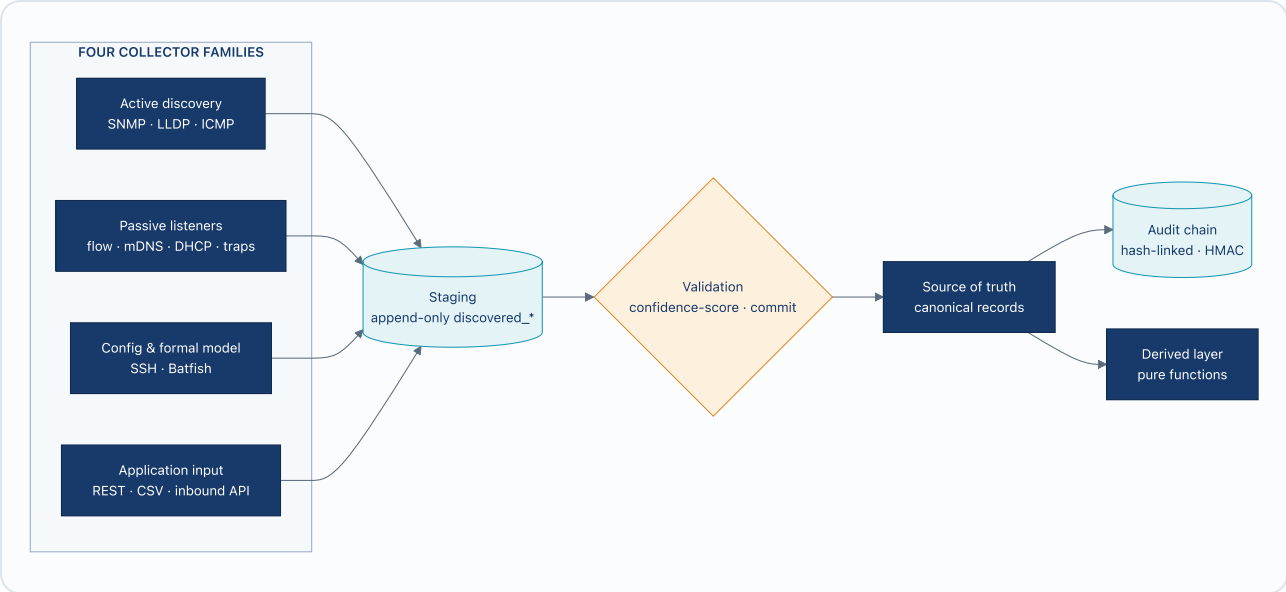

Figure 1. The collection map. Four read-only collector families feed one append-only staging store. Validation is the trust boundary into the source of truth and its hash-linked audit chain, and the derived layer reads from there. No collector writes the source of truth directly, apart from direct operator entry.

Two postures hold throughout. Active discovery (SNMP, LLDP, reachability) is the backbone. It runs on a timer, scheduled at `interval-ms` (default `300000`, 5 minutes) after a one-minute `initial-delay-ms`. The four passive listeners (flow, mDNS, DHCP, traps) are **dormant by default**: each is gated by its own `enabled` flag (default `false`) and binds no socket until an operator turns it on. Nothing a collector reads counts as truth on its own; it stays an observation until validation (§10) commits it.

2 The SNMP sweep: session & credentials

SNMP is the Simple Network Management Protocol, the standard way to read facts off network gear. The discovery worker (`crossconnect.discovery.enabled`, default `false`; set `source=snmp` to drive real SNMP rather than the seed source) queries each managed device over UDP/161 on a schedule. The session is built by `SnmpDiscoverySource` on `snmp4j 3.8.2`, and the per-device probe methods live in `SnmpProbe`. The session details a security reviewer or a network engineer wiring up an account will need:

Property	Value	Notes
Transport / port	`DefaultUdpTransportMapping`, UDP `161`	Outbound only; no agent installed, no port opened on the device.
SNMP versions	v2c (community) and v3 (USM)	v3 registers USM via `SecurityProtocols.addDefaultProtocols()`.
v3 auth	MD5, SHA, SHA-224/256/384/512	Mapped to snmp4j `AuthSHA` / `AuthHMAC*` protocol IDs.
v3 privacy	DES, 3DES, AES-128/192/256	Security level resolves to `AUTH_PRIV`, `AUTH_NOPRIV`, or `NOAUTH_NOPRIV`.

Property	Value	Notes
Read operation	scalar `GET`; table `GETBULK`	`TableUtils` with `maxNumRowsPerPDU = 20`; there is no `set` path, so it can only read, never write.
Timeout / retries	`1500 ms` per request, `1` retry	An unreachable device is counted and skipped; it never blocks the sweep.
Credential storage	AES-256-GCM, per-tenant	Community / v3 keys decrypted in memory only at probe time; never logged.
Credential pinning	`CredentialPinService`, per (tenant, device)	The credential that last worked is tried first on the next sweep, and the others are tried in order only if it fails.

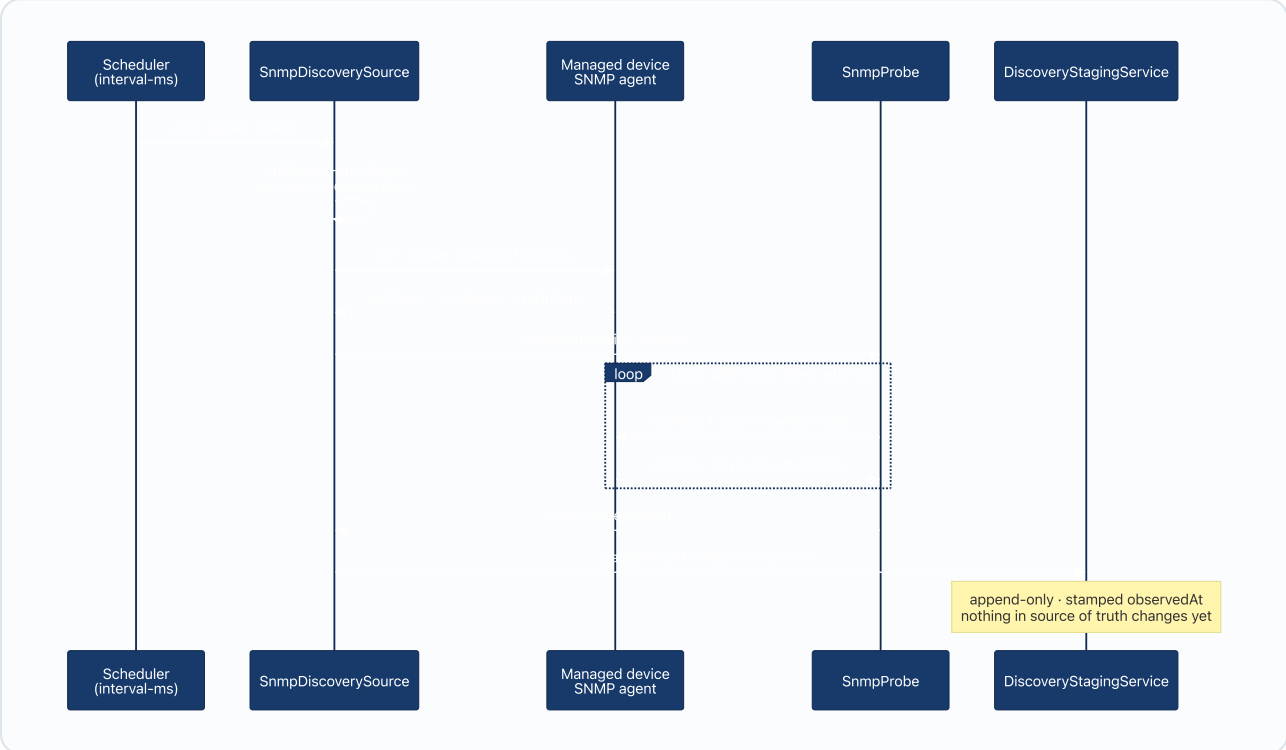

Figure 2. One probe sweep. The scheduler opens a tenant sweep, `SnmpDiscoverySource` pins a read-only credential and reads the system scalars, then `SnmpProbe` runs each MIB family best-effort over GETBULK. If a device does not support a given MIB, the probe returns an empty list and moves on, so one missing MIB never aborts the sweep. The decoded `SnmpSweepResult` is staged append-only.

3 SNMP inventory & topology MIBs

A MIB (Management Information Base) is the catalog of values a device exposes over SNMP. The inventory probes answer the basics: what is this device, what are its ports, and how is it wired. Each writes a single staging entity per natural key per sweep. The literal OID roots are in Appendix A.

Probe / MIB	Columns read	Staging → record
`probeSystem` RFC 1213 system	`sysDescr`, `sysObjectID`, `sysUpTime`, `sysName` (GET scalars); software version parsed from `sysDescr`	`DiscoveryFacts` → `Device` identity, platform, software, uptime. This is the one probe that always has to run.

Probe / MIB	Columns read	Staging → record
probeSerial ENTITY-MIB, RFC 4133	entPhysicalClass, entPhysicalSerialNum	Chassis serial (prefers class 3 = chassis). Committed to the Device record.
probeInterfaces IF-MIB, RFC 2863	ifDescr/ifName/ifAlias, ifType, ifMtu, ifSpeed + ifHighSpeed, ifPhysAddress (MAC), admin + oper status	DiscoveredInterface → Interface: every port, speed, MAC, state.
probeIfStack ifStackTable	ifStackStatus (higher / lower ifIndex pairs; sentinels at 0 dropped)	DiscoveredIfStack → port-channel (LAG) membership and sub-interface parents.
probeLldp LLDP-MIB, 802.1AB	lldpRemChassisIdSubtype/Id, lldpRemPortIdSubtype/Id, lldpRemPortDesc, lldpRemSysName/Desc	DiscoveredNeighbor → Cable links and the topology graph. A neighbour that maps to no known device is flagged as unmanaged.
probeVlans Q-BRIDGE, 802.1Q	dot1qVlanStaticName (VLAN id from row index)	DiscoveredVlan → Vlan.
probeEndpoints BRIDGE-MIB / IP-MIB	dot1dBasePortIfIndex, ipNetToMediaPhysAddress (ARP), dot1qTpFdbPort (per-VLAN FDB, legacy dot1dTpFdbPort fallback)	DiscoveredEndpoint (source=snmp) → which MAC and IP sit on which port and VLAN.
probeIps IP-MIB, RFC 1213	ipAdEntIfIndex, ipAdEntNetMask (mask → prefix length)	DiscoveredIp → IpAddress: host address and recovered prefix per interface.

4 SNMP routing, multicast, timing, environment

On the same session, discovery walks the more specialized MIBs below: routing, multicast, clock timing, and physical health. All are best-effort, so a device that does not support a given MIB is skipped for that one. Reachability is handled separately by an ICMP (ping) probe.

Probe / MIB	Columns read	Staging → meaning
probeBgp BGP4-MIB, RFC 4273	bgpLocalAs (scalar); per-peer bgpPeerState, bgpPeerAdminStatus, bgpPeerLocalAddr, bgpPeerRemoteAs, bgpPeerIdentifier	DiscoveredBgpPeer: peer AS, router-id, session state (idle…established), admin-up, established flags.
probeOspf OSPF-MIB, RFC 4750	ospfRouterId (scalar); per-neighbour ospfNbrIpAddr, ospfNbrRtrId, ospfNbrState	DiscoveredOspfNeighbor: neighbour id and adjacency state (down…full).

Probe / MIB	Columns read	Staging → meaning
`probeVrfs` MPLS-L3VPN, RFC 4382	`mplsL3VpnVrfRD` (VRF name decoded from length-prefixed row index)	`DiscoveredVrf` → the VRF / routing-domain model.
`probeMulticast` IGMP-MIB, RFC 2933	`igmpInterfaceQuerier` + `igmpInterfaceVersion` ; `igmpCacheSelf` (group from row index)	`DiscoveredQuerier` + `DiscoveredIgmpMembership` : the elected querier plus who is listening to which multicast group, which is evidence of a live Dante / NDI / AV stream.
`probePtp` PTPBASE-MIB (RFC 8173), CISCO-PTP fallback	`currentStepsRemoved` , `offsetFromMaster` (ns), parent GM identity, `priority1/2` , `clockClass` (6=GPS, 7=holdover, 248=free-run), port running state	`DiscoveredPtpClock` + `DiscoveredPtpPort` → PTP (Precision Time Protocol) clock-health quality for AV timing. It walks the standard tree first and falls back to the vendor tree.
`probeSensors` ENTITY-SENSOR, RFC 3433	`entPhySensorType/Scale/Precision/Value/OperStatus/UnitsDisplay` ; `entPhysicalName` for the label	`DeviceSensor` via `EntitySensorMapping` : temperature, fan, voltage, current, power, frequency, humidity, with ok / warning / critical status. Feeds Service readiness.
Reachability ICMP probe	Reachable or not, round-trip latency (ms) per sample	`DeviceReachability` (`source=probe`): the up/down history over time that sits behind Service readiness and device health.

> **PoE (Power over Ethernet).** `probePoe` walks POWER-ETHERNET-MIB (RFC 3621): `pethMainPsePower` (the power budget), `pethMainPseConsumptionPower` (measured watts), and oper status, writing one `DiscoveredPoe` per PSE group. This feeds the Rack power view and the building-presence preview.

5 Configuration over SSH & the formal model

Collecting a device's running configuration unlocks the formal-analysis layer, where CrossConnect reasons about the config itself. This collection is opt-in (`crossconnect.discovery.collect-config` , default `false`); the bean only loads when the flag is true, so a default deployment never opens an

SSH session. The collector is `SshConfigCollector` on `sshj 0.38.0`: it opens a read-only interactive shell, turns off paging, issues one read-only show command, captures the text, and exits. It never issues a `configure` or any other command that would change device state.

Vendor profile	Paging command	Show command (read-only)
Cisco IOS / IOS-XE	`terminal length 0`	`show running-config`
Cisco NX-OS	`terminal length 0`	`show running-config`
Juniper	`set cli screen-length 0`	`show configuration | display set | no-more`
Arista	`terminal length 0`	`show running-config`
Fortinet	`config system console / set output standard`	`show full-configuration`
Palo Alto	`set cli pager off / config-output-format set`	`show config running`
F5	`modify cli preference pager disabled`	`tmsh -q show running-config`

`VendorCliProfile.forVendor()` matches the device vendor slug without caring about case. The SSH connect and read timeouts are both `20` seconds. Host-key verification accepts the device key for a read-only collection session, and passwords are decrypted in memory only for the duration of that session.

Input	Source	What it becomes
Running config	`SSH`, post-sweep, opt-in	`DeviceConfig` (`kind=running`, `source=ssh`): captured state, normalized for diffing and fed to Batfish.
Intended / golden config	operator-set or imported	`DeviceConfig` (`kind=intended`): the baseline a device is checked against.
Configuration drift	computed	`ConfigDiff`: the lines added and removed between two captures (cosmetic differences filtered out), which is the signal behind golden-config drift.
Formal model	Batfish sidecar, from running configs	A vendor-neutral model that proves what can reach what, works out the effect of each ACL, infers topology from the config, and finds duplicate addressing. Read by `ReachabilityService`, `AclAnalysisService`, `ConfigTopologyService`.

6 Cloud-managed sources (vendor REST)

Some networks are run from a vendor's cloud dashboard rather than managed device by device. For those, CrossConnect pulls inventory from the vendor's dashboard API over HTTPS instead of walking SNMP. `CloudVendorSourceService` turns on per tenant once a base URL and bearer token are configured. It reads only the documented dashboard endpoints below, and the token is stored with AES-256-GCM and decrypted in memory only for the duration of the pull.

REST endpoint	What it reads	What it becomes
`/organizations/{org}/networks`	Network list under the organization	The set of networks to enumerate.
`/networks/{id}/appliance/vlans`	Appliance VLANs and subnets	`Vlan` + recovered prefixes for the network.
`/networks/{id}/appliance/firewall/l3FirewallRules`	Layer-3 firewall intent	Documented L3 policy for the network, surfaced for review.

> **Guarded against SSRF.** SSRF (server-side request forgery) is the trick of getting a server to call an address it should not. The HTTP client (JDK `HttpClient`) sets `followRedirects(NEVER)`, a 5-second connect and 10-second request timeout, and requires an HTTPS public host. Before it makes the call it resolves the target and rejects loopback, link-local, RFC1918, CGN (100.64/10), and IPv6 ULA (fc00::/7) addresses. A token is never sent to a private or rebindable target.

7 Passive listeners

Passive listeners query nothing. They listen for traffic summaries and the announcements gear already broadcasts on its own, and none of them looks inside a packet payload. Each listener is **off by default** (its `enabled` flag is `false`) and binds no socket until it is enabled and pinned to a tenant UUID, so the platform adds no network surface unless an operator turns one on.

Listener / class	Bind	What it captures	Default	Staging
NetFlow / sFlow `FlowListener`	`UDP 2055 / 6343`	5-tuples, byte and packet counts, exporting ifIndex (NetFlow v5/v9/IPFIX, sFlow v5)	OFF	`DecodedFlow` → `TrafficFlow`: top-talkers, per-application mapping, multicast / AV media flows.
mDNS `MdnsListener`	`mcast 224.0.0.251:5353`	Service type (`_dante._tcp`, `_ndi._tcp`, `_airplay._tcp`, `_rtsp._tcp`, any `_x._tcp/_udp`), instance name, TXT model, source IP	OFF	`DiscoveredMdnsService` → AV endpoint classification, all from the announcement and without touching a payload (10-min re-stage throttle per IP+type).
DHCP fingerprint `DhcpFingerprintListener`	`UDP 67`	Option-55 parameter list, option-60 vendor class, option-12 hostname, client MAC	OFF	`DiscoveredDhcpFingerprint` → a best guess at the device family (control box vs codec vs camera).
SNMP traps `SnmpTrapListener`	`UDP 162`	linkUp / linkDown, cold / warm start, auth failure, enterprise alarms (PSU, fan, temperature); v1 generic + v2c `snmpTrapOID`	OFF	`InboundObservation` (`source=snmptrap`) → classified events on the device timeline.

> The discovery sweep (§2–4) is the backbone and runs whenever it is enabled. These four listeners are extras on top of it. The push API in §8 feeds the same flow-ingestion path, so a collector can POST traffic summaries instead of exporting datagrams.

8 Application & integration input

Not every input comes off the wire. Operators and other systems hand records in directly. Manual entry is the one path that writes the source of truth without validation, because a person is treated as the authority. Everything else arrives as an observation and is validated like anything from a collector.

Input	Source	What it becomes
Manual entry	`REST` / UI	Operator-documented records (devices, cables, IPAM, VLANs, services). Written directly as *documented* truth.
Inbound event API	`POST /api/v1/inbound/event`	Claims pushed in by another system (header `X-CrossConnect-Tenant`; body `source/kind/summary/objectRef`). Staged as `InboundObservation`, returns `202`, and validated like any other observation.
Flow push	`TrafficFlowService.ingest()`	A collector POSTs flow summaries; merged into `TrafficFlow` keyed on (tenant, srcIp, dstIp, dstPort, protocol), counters accumulated.
Bulk import	`CSV` / REST	Records loaded in bulk into the source of truth, keyed so that re-importing the same file changes nothing.

Outbound paths (signed webhooks and SIEM / chat sinks) send data *out* rather than take it in, and they are covered in the Security & Architecture reference.

9 The staging entities

Every collector lands in a `discovered_*` staging table (plus the health and inbound tables). Each row carries `id`, `tenantId`, `observedAt`, and, where it applies to a device, a `deviceId`. Rows are only ever inserted, never updated, and the newest `observedAt` per natural key is the one that counts. The set written by the collectors in this document, with the distinctive fields each one carries:

Entity	From	Key fields
`DiscoveredInterface`	IF-MIB	ifIndex, ifName/Descr/Alias, ifType, mtu, speedBps, highSpeedMbps, macAddress, admin/oper status
`DiscoveredIfStack`	ifStackTable	higherIfIndex, lowerIfIndex
`DiscoveredNeighbor`	LLDP-MIB	localPortIndex, remoteChassis/Port subtype+id, remoteSysName, remoteSysDesc
`DiscoveredVlan`	Q-BRIDGE	vid, vlanName
`DiscoveredVrf`	MPLS-L3VPN	vrfName, rd

Entity	From	Key fields
`DiscoveredIp`	IP-MIB	ipAddress, prefixLen, ifIndex
`DiscoveredEndpoint`	FDB + ARP	mac, ip, vlan, ifIndex, source
`DiscoveredPoe`	POWER-ETHERNET	groupIndex, nominalPowerW, consumptionPowerW, operStatus
`DiscoveredBgpPeer`	BGP4-MIB	localAs, localAddr, peerAddr, peerAs, peerRouterId, state, adminUp, established
`DiscoveredOspfNeighbor`	OSPF-MIB	localRouterId, neighborIp, neighborRouterId, state, fullAdj
`DiscoveredQuerier`	IGMP-MIB	ifIndex, querierIp, igmpVersion
`DiscoveredIgmpMembership`	IGMP-MIB	groupAddress, ifIndex
`DiscoveredPtpClock` / `DiscoveredPtpPort`	PTPBASE / CISCO-PTP	domainNumber, gmIdentity, priority1/2, clockClass, offsetNs, stepsRemoved · portNumber, portState, source
`DiscoveredMdnsService`	mDNS listener	sourceIp, serviceType, instanceName, txtModel, sourceMac, vlan
`DiscoveredDhcpFingerprint`	DHCP listener	mac, ip, vlan, option55Csv, vendorClass, hostname
`DeviceSensor`	ENTITY-SENSOR	kind, label, value, unit, status
`DeviceReachability`	ICMP	reachable, latencyMs, source
`InboundObservation`	traps / inbound API	source, kind, objectRef, resolvedType/Key/Name, confidence, corroboratingSources, status

10 From signal to truth: validation

No matter where it came from, every collector above feeds the same pipeline under the same rules. Validation is the step that decides which observations become trusted records.

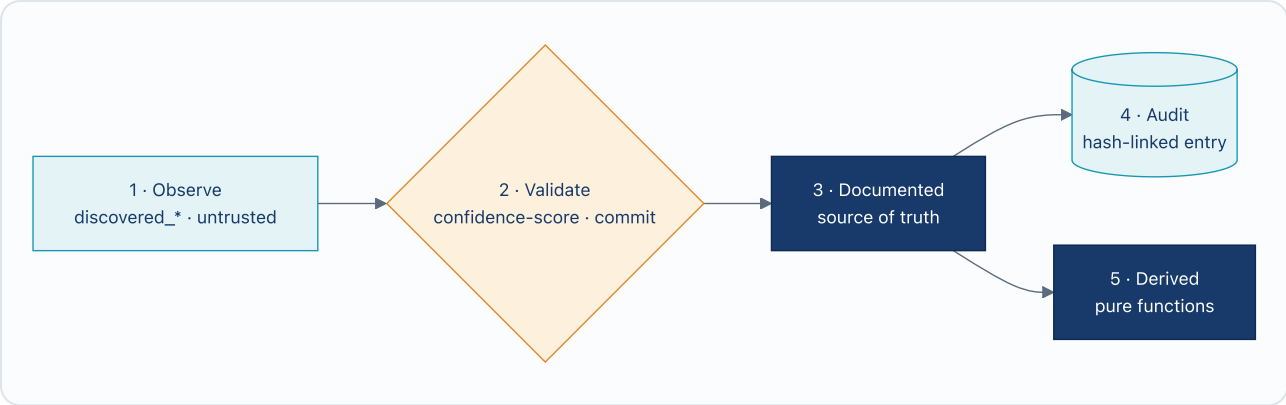

Figure 3. Data lifecycle. Confidence is earned by agreement between sources: two sources that agree score *Confirmed* / `high`, a single source is *Inferred* / `medium`, and an observation that maps to no known entity is *Unconfirmed* / `low` and flagged as a possible rogue. Sources only count as agreeing if they land within a 24-hour window. A commit is the only write path from observation to truth, and every one is audited.

- **Staging is append-only.** Each run writes a fresh row per natural key (for example device plus ifIndex), and the newest `observedAt` is the one that counts, so re-running a sweep is safe and never duplicates. Old staging rows are cleared automatically once they pass the retention window (`discovery.staging.retention-days`, default `14`) by a scheduled sweep.
- **Validation is the trust gate.** `ValidationService` confidence-scores staged observations based on how many sources agree: two independent sources agreeing within the 24-hour window score *Confirmed*, a single source scores *Inferred*, and one that resolves to nothing scores *Unconfirmed*. Once an observation earns enough confidence it is committed, and that is what creates the canonical record. An observation never overwrites a record silently.
- **The source of truth is hash-linked.** Every commit and edit publishes a `RecordChangeEvent` on the `EventBus` and is captured into a tamper-evident audit chain: `contentHash = SHA-256(tenantId · kind · occurredAt · actor · payload · previousHash)`, then signed with HMAC-SHA256. Because each entry chains to the one before it, the history of how a record reached its current state can be proven. A tenant's whole chain can be checked end to end (`EventAuditService.verifyChain(tenantId)`), and it is kept on its own retention policy (`audit.retention-days`, default `90`).
- **The derived layer reads, it does not collect.** Data-quality, compliance, readiness, reachability, and the AV and presence views are all computed purely from the source of truth, and they store no new facts of their own.

Cross-cutting rule	How it applies to every input
Tenancy	Every staged row and every record belongs to one tenant. The tenant is the line that keeps customers separate, and inbound APIs require a tenant header.
No payload inspection	Collectors read switch-derived signals, announcements, and configuration text only. No collector captures or parses packet payloads, and there is no SPAN or mirror feed.
Default-off listeners	The four passive listeners bind no socket until their `enabled` flag is set and a tenant is pinned. Active discovery is the only collector that runs as soon as it is enabled.
Idempotency	Staging and imports key on a natural key, so collecting the same thing twice converges on one row instead of duplicating it.

Cross-cutting rule	How it applies to every input
Retention	Append-only staging is dropped past its window (default 14 days); the audit chain is retained on its own policy (default 90 days), preserving link integrity.

11 Worked example: one link, wire to map

Take one fact and follow it the whole way through. Switch `acc-sw-3` port `Gi1/0/14` is patched to `core-sw-1` port `Gi1/0/1`. Every signal in this document goes through the same five moves, **read, stage, validate, commit, prove**. This traces one of them in full, with the exact OIDs, staging tables, services, and records.

1. **Trigger.** The discovery scheduler (`interval-ms` , default about 5 minutes) opens a tenant sweep and, for each documented device, has `SnmpDiscoverySource` build a read-only snmp4j session to the management IP on UDP/161 with the pinned credential. No agent is installed and no port is opened on the device.
2. **Read the wire.** On `acc-sw-3` the sweep walks IF-MIB and LLDP-MIB. From `ifTable` / `ifXTable` it reads `ifName` (`Gi1/0/14`), speed, MAC, and admin and oper status. From `lldpRemTable` it reads, on local port `Gi1/0/14` , the remote `lldpRemSysName` (`core-sw-1`) and `lldpRemPortId` (`Gi1/0/1`), with chassis and port subtypes. When the sweep later reaches `core-sw-1` it reads the mirror-image neighbour, so the one link is observed independently from both ends.
3. **Parse.** `SnmpProbe` decodes the raw varbinds into a `SnmpSweepResult` : a list of `SnmpInterfaceFact` and `SnmpLldpNeighbor` records, each stamped with `tenantId` , the resolved `deviceId` , and `observedAt` .
4. **Stage, append-only.** `DiscoveryStagingService` inserts a `DiscoveredInterface` row for `acc-sw-3 / Gi1/0/14` and a `DiscoveredNeighbor` row (local `Gi1/0/14` to remote `core-sw-1 / Gi1/0/1`). Rows are inserted, never updated; the newest `observedAt` per natural key is operative, so the next sweep is idempotent. Nothing in the source of truth has changed yet: the link is observed, not documented.
5. **Validate, the trust gate.** `ValidationService` confidence-scores the staged neighbour. It resolves both endpoint names, and since both are managed devices this is a candidate cable between two known ports. It checks whether a `Cable` already records the pair; if not, the link is queued to commit. Because both switches reported the same adjacency, the link scores **Confirmed**. (A one-sided LLDP sighting would score *Inferred*, and a neighbour whose name resolves to no device would score *Unconfirmed* and be flagged as a possible rogue.) The link then waits in the validation queue with its evidence and score, still outside the source of truth.
6. **Commit.** Once the link has earned enough confidence, `commit()` is the only write path from observation into truth. It find-or-creates the two `Interface` endpoints and writes one `Cable` joining `acc-sw-3:Gi1/0/14` and `core-sw-1:Gi1/0/1` , stamping `discoveredAt` . The observation is now a record.
7. **Record and prove.** The write publishes a `RecordChangeEvent` on the `EventBus` . The audit plugin captures it into the tamper-evident chain, `contentHash = SHA-256(tenantId · kind · occurredAt · actor · payload · previousHash)` , HMAC-signed and linked to the previous entry for the tenant. The link is provable: who added it, when, and that the record has not been altered since.

8. **Downstream updates, at no extra cost.** Because the derived layer is computed purely from the source of truth, the new `Cable` immediately shows up on the topology graph and the network map; clears the data-quality "undocumented link" finding the LLDP sighting had raised while it was still staged; joins the Batfish topology input, so reachability checks now traverse it; and becomes citable by the AI assistant, which can reference the `Cable` record and its audit entry by primary key.

One LLDP value, read read-only off two switches, became a Confirmed, operator-approved, cryptographically recorded cable the whole platform now reasons over.

A Appendix, OID reference by probe

The literal OID roots each SNMP probe walks (GET for single values, GETBULK for tables). No state-changing OID `set` is ever issued, so collection can only read.

Probe	OID root(s)	Object
probeSystem	`1.3.6.1.2.1.1.1/.2/.3/.5.0`	sysDescr, sysObjectID, sysUpTime, sysName
probeSerial	`1.3.6.1.2.1.47.1.1.1.1.5 / .11`	entPhysicalClass, entPhysicalSerialNum
probeInterfaces	`1.3.6.1.2.1.2.2.1.*`, `1.3.6.1.2.1.31.1.1.1.{1,15,18}`	ifTable + ifXTable (name, alias, highSpeed)
probeIfStack	`1.3.6.1.2.1.31.1.1.3.1.3`	ifStackStatus
probeLldp	`1.0.8802.1.1.2.1.4.1.1.{4..10}`	lldpRemTable
probeVlans	`1.3.6.1.2.1.17.7.1.4.3.1.1`	dot1qVlanStaticName
probeEndpoints	`1.3.6.1.2.1.17.1.4.1.2`, `1.3.6.1.2.1.4.22.1.2`, `1.3.6.1.2.1.17.7.1.2.2.1.2` / `17.4.3.1.2`	basePortIfIndex, ipNetToMedia, dot1q/dot1d FDB
probeIps	`1.3.6.1.2.1.4.20.1.2 / .3`	ipAdEntIfIndex, ipAdEntNetMask
probePoe	`1.3.6.1.2.1.105.1.3.1.1.{2,3,4}`	pethMainPsePower / OperStatus / ConsumptionPower
probeBgp	`1.3.6.1.2.1.15.2.0`, `1.3.6.1.2.1.15.3.1.{1,2,3,5,9}`	bgpLocalAs, bgpPeerTable
probeOspf	`1.3.6.1.2.1.14.1.1.0`, `1.3.6.1.2.1.14.10.1.{1,3,6}`	ospfRouterId, ospfNbrTable
probeVrfs	`1.3.6.1.2.1.10.166.11.1.2.2.1.4`	mplsL3VpnVrfRD
probeMulticast	`1.3.6.1.2.1.85.1.1.1.{3,4}`, `1.3.6.1.2.1.85.1.2.1.4`	igmpInterfaceVersion/Querier, igmpCacheSelf
probePtp	`1.3.6.1.2.1.241.1.2.*` (std), `1.3.6.1.4.1.9.9.760.1.2.*` (fallback)	PTPBASE clock/parent/port datasets

Probe	OID root(s)	Object
probeSensors	`1.3.6.1.2.1.99.1.1.1.{1..6}`, `1.3.6.1.2.1.47.1.1.1.1.7`	entPhySensor*, entPhysicalName

B Appendix, collector configuration reference

What gets collected is driven by configuration. Representative properties from `application.yml` follow. The defaults are deliberately quiet, with almost everything off until an operator turns on what they need:

Collector	Property	Default
Discovery worker	`crossconnect.discovery.enabled`	`false`
Discovery source	`crossconnect.discovery.source`	`stub` (set `snmp` for real SNMP)
Sweep interval	`crossconnect.discovery.interval-ms`	`300000` (5 min)
Initial delay	`crossconnect.discovery.initial-delay-ms`	`60000` (1 min)
Config (SSH) collection	`crossconnect.discovery.collect-config`	`false`
SSH timeout	`crossconnect.discovery.ssh.timeout-seconds`	`20`
mDNS listener	`crossconnect.discovery.mdns.enabled` · `.port` · `.group`	`false` · `5353` · `224.0.0.251`
DHCP listener	`crossconnect.discovery.dhcp.enabled` · `.port`	`false` · `67`
Flow listener	`crossconnect.integrations.flow.enabled` · `.netflow-port` · `.sflow-port`	`false` · `2055` · `6343`
Trap listener	`crossconnect.integrations.snmptrap.enabled` · `.port`	`false` · `162`
Inbound event API	`crossconnect.integrations.inbound.enabled`	`false`
Credential master key	`CROSSCONNECT_CREDENTIALS_AES_KEY` / `…_KEY_COMMAND` / `…_KEY_FILE`	resolved in priority order; fails closed in prod if unset
Staging retention	`crossconnect.discovery.staging.retention-days`	`14`
Audit retention	`crossconnect.audit.retention-days`	`90`