CrossConnect
CONNECT. MONITOR. OPTIMIZE.
BY Cybr IQ

Vendor Support Reference

For every network and AV vendor CrossConnect handles: what we collect, how we reach it, how we parse it, what models it, and what we do with the result. Written plainly, so an engineer can see exactly how their estate is treated, vendor by vendor.

Audience: network engineers, security reviewers, AV integrators

Scope: acquisition, parsing, modeling, and use, per vendor, with each capability's maturity

Document: engineering reference, 21 June 2026

Contact: contact_us@cybriq.io

0 How to read this document

CrossConnect treats a device by what it can actually do with that device's configuration, not by brand loyalty. Three things decide the treatment: how we get the configuration (SSH, a cloud API, or SNMP), what can parse it (a formal model, our own parser, or none), and therefore what analysis we can offer. Every vendor below is labelled with its maturity so nothing is oversold.

PRODUCTION shipped and working end to end today

BUILT, WIRING EXPANDING the parser or client exists; full surfacing is in progress

ROADMAP designed and prototyped, not yet built into the product

The single source of truth for a vendor's treatment is the `VendorSupport` capability matrix in the code: for each vendor it records how config is captured (`SSH` , `CLOUD_API` , or `UNSUPPORTED`), whether the formal engine can model it (`SUPPORTED` / `UNSUPPORTED`), and an operator-facing note when support is limited. This document spells that matrix out.

1 The four ways we treat a vendor

A vendor falls into one of four tiers. The tier sets what analysis is possible, and the code decides it, not a sales sheet.

Formally analyzed

Config is captured over SSH and parsed by the Batfish formal model. We can prove reachability, validate access rules, trace a breaking change (the Network Black Box), and grade config correctness. Cisco, Arista, Juniper, Fortinet, Palo Alto, F5.

Config-level

Config is captured over SSH and parsed by our own parser, alongside SNMP facts. We analyze segmentation, VLANs, multicast, access-rule intent, PoE, and hardening, but not formal cross-device reachability. Netgear, Ubiquiti.

Cloud-managed

There is no on-device running-config; configuration lives in a vendor cloud and is pulled over a REST API, then analyzed at the config level. Meraki, Ubiquiti UniFi.

Capture-only

The running-config can be captured, stored, searched, diffed, and drift-checked over SSH, but nothing parses it, so no model-based analysis is offered. Extreme, and any vendor we do not yet recognize.

Two cross-cutting layers sit alongside these tiers and apply to every vendor: **SNMP** (§7) gives interfaces, VLANs, neighbors, multicast, timing, and sensors regardless of brand, and the **AV endpoint** layer (§8) types audio-visual gear that often speaks neither SSH nor SNMP.

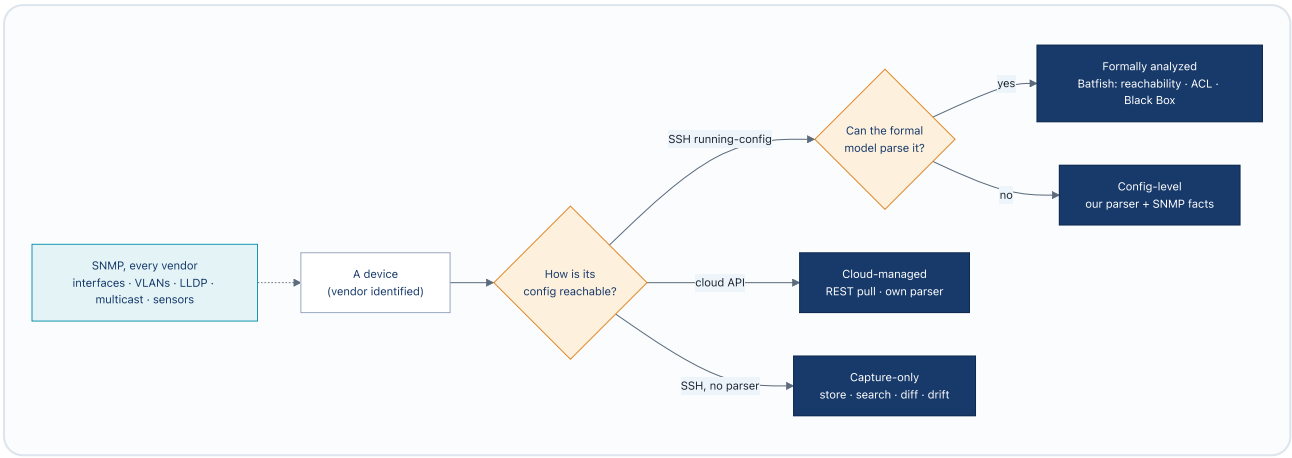

Figure 1. How a device is routed to a tier. Acquisition and parseability decide the tier. Formal analysis is reserved for vendors the formal model can parse; everything else is captured and analyzed at the level it supports. SNMP runs in parallel for all vendors.

2 How we acquire configuration

Every acquisition path is read-only. We never push configuration to a device. There are four ways data comes in.

Path	Transport	What it fetches	How
SSH running-config	`22/TCP` SSH (sshj)	The device's running configuration as text	`SshConfigCollector` opens an interactive shell, disables paging, and runs the vendor's read-only show command (Appendix A). Opt-in.
Cloud REST API	`443/TCP` HTTPS	VLANs, firewall rules, switch-port and device inventory	`CloudVendorSourceService` calls the vendor dashboard with an encrypted API key; the target URL is SSRF-guarded (public HTTPS only).
SNMP	`161/UDP` SNMP	Interfaces, VLANs, LLDP, IP, multicast, PTP, sensors	`SnmpProbe` read-only GET / GETBULK walks (snmp4j). Vendor-agnostic. On by default.
Link-local discovery	mDNS / DHCP listen	AV service announcements, device fingerprints	Passive listeners on the local segment (§8). Off by default.

The SSH path is the one that varies by vendor: each vendor needs a different paging-disable and a different show command, and the captured text needs to be tagged with its format so the right parser is selected. That per-vendor profile is the `VendorCliProfile`, listed in full in Appendix A.

3 Formally analyzed vendors PRODUCTION

For six vendor families we capture the running-config over SSH and parse it with the **Batfish formal model**. Batfish turns each vendor's configuration into one vendor-independent model of how the network forwards and filters traffic. From that model we can prove a path, validate an access rule across vendors, and trace the exact change that broke something. Each config is tagged with its vendor format before it enters the model so the correct grammar is used.

Vendor	How we get it (SSH)	How we parse it	What we use it for
Cisco IOS / IOS-XE	`terminal length 0`, `show running-config`	Batfish (IOS grammar, format auto-detected)	Reachability proofs, access-rule (ACL) validation, the Network Black Box root-cause trace, config-correctness grading, golden-config drift. Hardening heuristics on Cisco, Arista, Juniper.
Cisco NX-OS	`terminal length 0`, `show running-config`	Batfish (NX-OS grammar)	
Arista EOS	`terminal length 0`, `show running-config`	Batfish (EOS grammar)	
Juniper JunOS	`set cli screen-length 0`, `show configuration | display set | no-more`	Batfish (set-format grammar)	
Fortinet FortiOS	console setup, `show full-configuration`	Batfish (security-policy model)	
Palo Alto PAN-OS	`set cli config-output-format set`, `show config running`	Batfish (set-format model)	
F5 BIG-IP	`tmsh -q show running-config`	Batfish (BIG-IP model)	

We state coverage honestly. The formal model covers the six families above. If a path crosses a device whose config we captured but the formal model cannot parse (a config-level or capture-only vendor), the Black Box and reachability results carry a coverage note naming those devices, because a proof cannot be completed across a hop the model does not understand. Firewall vendors (Fortinet, Palo Alto, F5) are formally modeled for reachability and policy, but the rule-based hardening heuristics run on the routing-and-switching families (Cisco, Arista, Juniper).

4 Config-level vendors (our own parsers) BUILT, WIRING EXPANDING

Some vendors the formal model does not parse, but their configuration is still worth reading. Rather than wait for a formal grammar (which can take many months), we wrote small, focused parsers that pull the facts that matter for segmentation, multicast, and hardening. These devices are modeled as Layer-2 bridges or opaque Layer-3 hops inside the topology; cross-device reachability proofs still run on the formally analyzed core.

Vendor	How we get it	How we parse it	What we extract & use
Netgear FASTPATH (M4250 / M4300 / M4350)	SSH running-config + SNMP	`FastpathConfigParser` (our own, no external grammar)	Interfaces, VLANs and switchport mode, access rules (ACLs), IGMP snooping and querier, PoE, static routes. Used for segmentation checks, multicast validation, access-rule intent, PoE budget, hardening, and AV-edge topology.
Ubiquiti EdgeSwitch	SSH running-config + SNMP	`FastpathConfigParser` (same family as Netgear, reused)	Interfaces, VLANs, IGMP, access rules, PoE. Same segmentation, multicast, and hardening checks.
Ubiquiti EdgeRouter (EdgeOS)	SSH configuration + SNMP	`VyattaConfigParser` (our own, brace-tree and set-form)	Interfaces and VLAN sub-interfaces, firewall rule sets (access-rule analog), static routes, NAT. Used for firewall intent, Layer-3 edge topology, and hardening.

The `VendorSupport` matrix ships and classifies these vendors today, and the parsers above are written and unit-tested. What remains is surfacing each parser's output into every analyzer view, which is why this section is marked *built, wiring expanding* rather than fully production.

5 Cloud-managed vendors

Cloud-managed gear keeps its configuration in a vendor cloud, so there is no SSH running-config to capture. CrossConnect pulls the configuration over the vendor's REST API instead, with an encrypted API key, and analyzes it at the config level (the same segmentation and access-rule intent checks, no formal reachability).

Vendor	How we get it	How we parse it	What we extract	Status
Cisco Meraki	Dashboard REST API over HTTPS (encrypted key, SSRF-guarded), dormant until configured	`MerakiJson` maps the JSON into `MerakiFacts`	VLANs (id, name, subnet), Layer-3 firewall rules, switch-port profiles (VLAN mode, PoE), device inventory	BUILT, WIRING EXPANDING

Vendor	How we get it	How we parse it	What we extract	Status
Ubiquiti UniFi	UniFi Controller REST API over HTTPS (encrypted key)	`UniFiJson` maps the controller JSON, reusing the Meraki fact types	VLAN-bearing networks and L3 firewall rules	BUILT, WIRING EXPANDING

Both follow the same shape. The API client and the JSON parser are built for each: `MerakiJson` for the Meraki Dashboard, and `UniFiJson` for the UniFi controller (it parses the controller's networks and firewall rules into the same fact types). For both, the configuration gate and the wiring of those facts into the analyzer views are being completed.

6 Capture-only & unrecognized vendors PRODUCTION

Anything we can reach over SSH but cannot parse is still useful as text. We capture the running-config, store it, let you search and diff it, and check it for drift against an approved baseline. We just do not claim a model-based analysis we cannot back up.

- **Extreme (EXOS)** is the named capture-only vendor: config is captured, stored, searched, diffed, and drift-checked, with no formal or config-level parsing.
- **Any unrecognized vendor** is treated as SSH best-effort: we attempt a capture, but we do not assert formal analysis, config grading, or hardening until that vendor is properly supported. The operator-facing note says so plainly.

7 SNMP, across every vendor PRODUCTION

SNMP is the one path that works the same way on every brand, and it runs by default. It reads standard MIBs (the structured data tables a device exposes) for the facts that do not need a config parser, plus a few vendor-specific tables where the standard ones fall short.

Fact	Standard MIB	Applies to
Interfaces (name, speed, MAC, status)	IF-MIB	All vendors
VLAN membership and names	Q-BRIDGE-MIB	All VLAN-capable vendors
LLDP neighbors	LLDP-MIB	All LLDP-capable vendors
IP addresses and masks	IP-MIB	All vendors
Multicast (IGMP querier, group membership)	IGMP-MIB	All vendors; parsed by `MulticastSnmpParser`
Precision timing (PTP)	PTP MIBs, with a Cisco fallback	PTP-capable gear; parsed by `PtpSnmpParser`
Environment sensors	ENTITY-SENSOR-MIB	All vendors
Identity (model, serial, vendor)	MIB-II + ENTITY-MIB	All vendors; used to detect the vendor and pick the right SSH profile

Where a standard MIB is not enough, we read a vendor-specific one: for example the Cisco PTP extensions when a device does not populate the standard PTP table, and a Netgear-private power table for PoE budget on FASTPATH switches. The enterprise number in a device's system identity is also how we recognize cloud-managed gear (for example Meraki) before any API is connected.

8 AV endpoint vendors

Audio-visual gear, codecs, displays, cameras, microphones, signal processors, often speaks neither SSH nor SNMP. CrossConnect types it from the signals it does give off, and for some roles can poll the device's own management port to confirm it is live. The vendor matters here in two ways: the MAC-address vendor (OUI) is a strong hint at what a device is, and the management protocol is vendor-family specific.

Vendor identity to role

`AvOuiProfile` maps a MAC-address vendor to a likely AV role. AV-only vendors (Audinate/Dante, Shure, Crestron, AMX, Poly, Biamp, QSC, Barco, NEC, Axis, and others) are a strong vote on their own. Multi-market vendors (Cisco, Sony, Panasonic, Apple, Microsoft) only count as a vote when a second signal agrees, so an Apple device is not called a display until something like an AirPlay announcement corroborates it.

Live management-port probes

Where a role has a standard management protocol, a probe confirms the device is real and reports its state, live first, falling back to the classified record when a probe is not available or not enabled.

Role	Protocol / port	Probe	Status
Display	PJLink, `4352/TCP`	`PjLinkProbe` (power, input, name)	PRODUCTION
Signal processor (DSP)	Q-SYS QRC, `1710/TCP`	`QsysProbe` (run-state, loaded design)	PRODUCTION
Codec	SIP, `5060/UDP`	`SipProbe` (liveness, vendor/model)	BUILT
Camera	ONVIF, `80/TCP`	`OnvifProbe` (device info, liveness)	EXPERIMENTAL, DORMANT

The classifier also fuses passive signals: mDNS service types (`_dante` , `_qsys` , `_airplay`), LLDP model strings, DHCP fingerprints, and observed media flows (Dante, AES67, NDI, Q-SYS). The camera probe is part of the experimental Peek-a-Boo preview and is off by default (`crossconnect.peekaboo.probe-enabled=false`) until camera credentials are supplied; the display and DSP probes are part of the shipped AV endpoint modules. Full detail is in the AV sections of the Capability Guide and the Experimental Features explainer.

Figure 2. AV vendor to role to probe. The MAC vendor and passive signals classify a device into an AV role with a confidence label; where a standard management protocol exists, a live probe confirms it. The camera (ONVIF) probe is the experimental Peek-a-Boo path and stays dormant until enabled.

9 Occupancy & wireless vendors PRODUCTION

For room occupancy and wireless health, CrossConnect integrates with the wireless vendor's cloud, with an encrypted API token, and reads placement and client-count telemetry. This feeds occupancy analytics, not network-core modeling.

Vendor	How we get it	What we pull	What we use it for
Juniper Mist	Mist cloud API (encrypted token)	AP placement, zone geometry, per-zone client counts over time	Room occupancy, utilization, presence
Cisco Catalyst Center	Catalyst Center REST API	AP inventory, RF health, client association and location, assurance alerts	Occupancy, AP and RF health, assurance
Cisco Spaces	Cisco Spaces location telemetry	Client location by zone and floor	Occupancy and location analytics

10 What ships today vs what is expanding

Because customers ask exactly this, here is the honest split. Nothing below is a future promise dressed as a present feature.

Area	Shipped & working	Built, wiring expanding	Roadmap (designed)
Formal analysis	Cisco, Arista, Juniper, Fortinet, Palo Alto, F5 (SSH + Batfish)	n/a	Further firewall and switch families
Config-level parsing	The `VendorSupport` classification + capture	Netgear & Ubiquiti EdgeSwitch (FASTPATH parser), EdgeRouter (EdgeOS parser): parsers built, full analyzer/UI wiring	n/a
Cloud-managed	n/a	Meraki Dashboard and Ubiquiti UniFi: API client + JSON parser built	n/a

Area	Shipped & working	Built, wiring expanding	Roadmap (designed)
Capture-only	Extreme and unrecognized vendors (store, search, diff, drift)	n/a	n/a
SNMP	Interfaces, VLANs, LLDP, IP, multicast, PTP, sensors, identity	Netgear-private PoE table	n/a
AV endpoints	OUI + mDNS classification; Display (PJLink) and DSP (Q-SYS) probes	Codec (SIP) probe; the unified classifier engine and AV views	External AV inventory ingest (control-plane platforms)
Occupancy	Mist, Cisco Catalyst, Cisco Spaces	n/a	n/a

A Appendix, SSH command profile per vendor

The per-vendor SSH profile (`VendorCliProfile`) sets the paging-disable command and the read-only show command CrossConnect runs to capture a running-config. All commands are read-only; no configuration command is ever issued.

Vendor	Disable paging	Capture command
Cisco IOS / IOS-XE	`terminal length 0`	`show running-config`
Cisco NX-OS	`terminal length 0`	`show running-config`
Arista EOS	`terminal length 0`	`show running-config`
Juniper JunOS	`set cli screen-length 0`	`show configuration | display set | no-more`
Fortinet FortiOS	console output setup	`show full-configuration`
Palo Alto PAN-OS	`set cli config-output-format set`	`show config running`
F5 BIG-IP	`modify cli preference pager disabled`	`tmsh -q show running-config`

Vendors without a dedicated profile (for example Netgear, Ubiquiti EdgeSwitch, Extreme) are captured with a best-effort show command and routed to the parser or capture-only handling described above.

B Appendix, vendor capability matrix

The shipping `VendorSupport` catalog, ordered most-capable first. *Capture* is how config is obtained; *Formal* is whether the Batfish model can parse it.

Vendor family	Capture	Formal model	Analysis tier
Cisco	SSH	Supported	Formal
Arista	SSH	Supported	Formal
Juniper	SSH	Supported	Formal
Fortinet	SSH	Supported	Formal
Palo Alto	SSH	Supported	Formal
F5	SSH	Supported	Formal
Netgear	SSH	Unsupported	Config-level (own parser + SNMP)
Ubiquiti	SSH (UniFi: cloud API)	Unsupported	Config-level / cloud
Extreme	SSH	Unsupported	Capture-only
Meraki	Cloud API	Unsupported	Cloud (config-level)
Unrecognized	SSH (best effort)	Unsupported	Capture-only

CrossConnect by CybrIQ · Vendor Support Reference · Technical reference · 21 June 2026 · Tiers and statuses reflect the shipping `VendorSupport` matrix and the code as of this date; items marked roadmap are designed, not yet built. · contact_us@cybriq.io