Week 2 Reference Card · How Jamf Pro Operates

Admin Crossover · Your First 30 Days as a Jamf Pro Administrator, Part 2 · admincrossover.dev

New to Apple management from a Windows/AD world? Start here. Jamf does the same jobs you know — it just splits them across two channels and swaps the vocabulary. This card is read-only reference: nothing here changes production.

The translation: what you know → what Jamf calls it

Windows / Active Directory	Jamf Pro equivalent	What's different
Group Policy Object (GPO)	Configuration Profile	One payload per profile; updating replaces the whole profile (no delta merge)
SCCM / Intune agent	The **jamf** binary + Apple **MDM** framework	Two separate channels — see below
AD security group / OU	Smart Group (dynamic) / Static Group	Smart Groups re-evaluate on inventory update, not instantly
GPO link / OU scope	Scope (targets + exclusions) on a policy or profile	Set per-object, not by directory location
gpupdate /force	`sudo jamf policy`	Triggers a recurring-check-in run on demand
gpresult / RSoP	Policy Logs + `jamf policy -verbose`	Per-device history lives on the computer record
WSUS / SCCM updates	Patch Management / App Installers	Title-based; App Installers are Jamf-maintained

Two channels — the one idea that unlocks everything

MDM (Apple push channel): installs Configuration Profiles and runs device commands (lock, wipe, inventory request, restart). Apple-authenticated, works even when the agent is unhealthy. Only MDM can install profiles.

The jamf agent (binary on the Mac): runs Policies and Scripts, submits inventory (`recon`), and handles Self Service. Only the agent runs scripts and multi-step policies. If a profile won't apply, suspect MDM/APNs; if a script or policy won't run, suspect the agent/check-in.

The check-in cycle — what runs, and when

Cycle	Channel	Default cadence	What it does
Recurring check-in	jamf agent	Every 15 min (configurable)	Runs policies scoped to the *recurring check-in* trigger
MDM check-in	Apple push	On push / periodically	Delivers profiles & device commands
Inventory submission (recon)	jamf agent	~Once daily / on trigger	Updates what Jamf knows; **Smart Groups recalc from this**

Why a device "isn't getting" something: most often its inventory is stale, so it hasn't fallen into the Smart Group the policy is scoped to. Force a fresh submission with **`sudo jamf recon`**, then re-check the group.

Policy execution sequence (what happens on a check-in)

1. A **trigger** fires (recurring check-in, or a custom event). 2. The jamf binary checks in with the server. 3. The server evaluates **scope** (is this device a target, and not excluded?). 4. **Frequency** is checked (e.g. *Once per computer* already run → skip). 5. The policy runs its payloads (before / during / after). 6. Result is written to the **policy log** (Completed / Failed / Pending).

Week 2 Reference Card · Commands, Logs & Troubleshooting

*Admin Crossover · First 30 Days, Part 2 · Run these on a **test** Mac — reading state is safe; nothing here changes production*

The jamf binary — everyday commands

Command	What it does
`sudo jamf policy`	Run all recurring-check-in policies now (like gpupdate)
`sudo jamf policy -event`	Run policies attached to a custom trigger
`sudo jamf policy -verbose`	Same, with full output — the go-to for troubleshooting
`sudo jamf recon`	Submit fresh inventory now (recalculates Smart Groups)
`sudo jamf manage`	Re-apply the management framework / MDM profile
`sudo jamf mdm`	Re-establish the device's MDM enrollment
`sudo jamf checkJSSConnection`	Test that this Mac can reach the Jamf Pro server
`sudo jamf about`	Show the server URL and version this Mac is talking to
`jamf help`	List every jamf binary verb

Where the logs live

Location	What's in it
`/var/log/jamf.log`	Agent + policy activity on the Mac — your first stop on the device
`/var/log/install.log`	Package install detail (pair with a failed install policy)
`log show --predicate 'process == "jamf"' --last 1h`	Unified-log view of recent jamf activity
`Console (policy)` → `Logs button on the policy`	Server-side result across all scoped devices
`Computer record` → `History` → `Policy Logs`	Full policy history for ONE device
`Computer record` → `Management Commands`	Pending / completed / failed MDM commands

Smart Group / “why isn't this device getting it?” checklist

- [ ] Is the device actually **in** the group? Computer record → Groups & Policies.
- [ ] Do the group **criteria** really match this device? Watch and/or logic and exact values (OS version, dept, model).
- [ ] Has the device **submitted inventory** since the criteria changed? Smart Groups recalc on recon — run `sudo jamf recon`.
- [ ] Is the policy/profile **scoped** to that group as a **target** (and not caught by an **exclusion**)?
- [ ] Is the **trigger** right, and is **frequency** blocking it? (“Once per computer” already ran = it won't run again.)
- [ ] Check the **policy log** for this device: Completed, Failed, or Pending? Read the Failed detail.
- [ ] Still stuck? **Flush** the failed policy log (Computer record → History → Policy Logs) so it can retry on next check-in.
- [ ] Profile, not policy? A profile needs **MDM/APNs** healthy — confirm the device still shows as managed.

End of Week 2 you should be able to read `jamf.log` and correlate an entry to a policy run you watched in the console, and have one Static Test Group ready for controlled evaluation. Read-only reference — part of the First 30 Days series at **admincrossover.dev**.