AUGMENTT

The Intune Policy Playbook (MSP Edition)

How to Design, Deploy, and Scale Intune Policies Without Breaking Client Environments

AUGMENTT

Table of Contents

Introduction

Why This Playbook Exists

Intune is powerful. But it was built with for businesses focusing on one environment — not those of us who are thinking about multi-tenant management.

That's why it's also one of the fastest ways for MSPs to create chaos inside a client environment if not deployed carefully.

Most MSPs don't struggle because Intune is too technical. They struggle because:

- ❌ They deploy policies reactively.
- ❌ They mix policy types without understanding overlap.
- ❌ They assign everything to "All Devices."
- ❌ They build per-client snowflakes instead of repeatable frameworks.
- ❌ Switching between Microsoft portals is time-consuming.

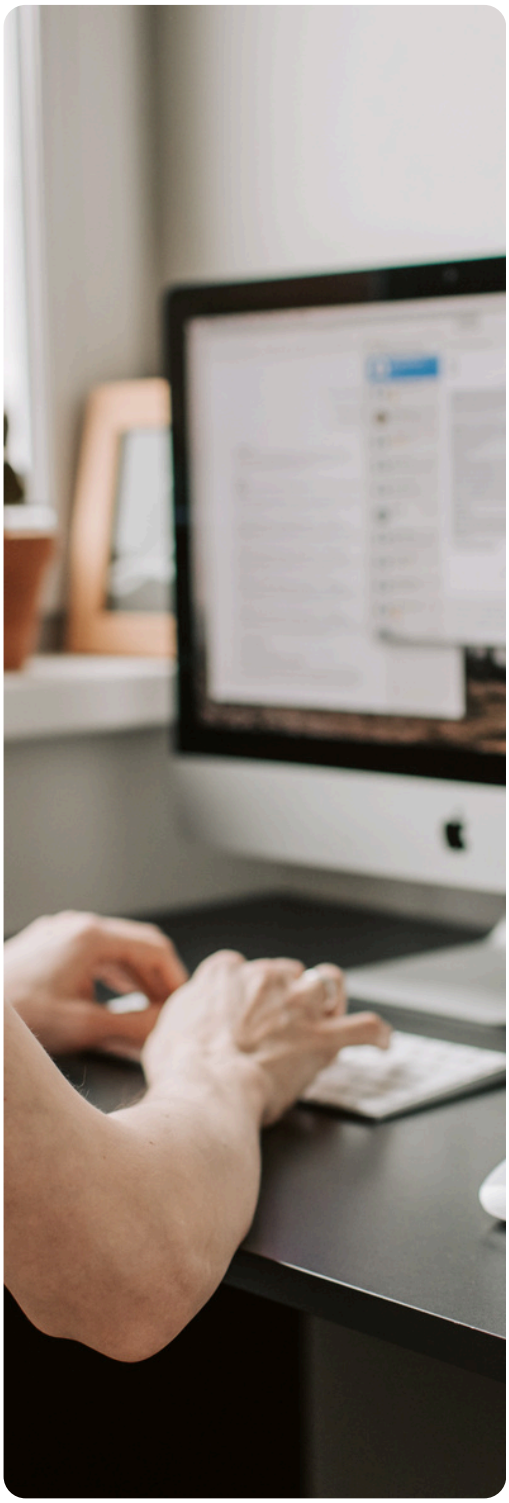
This playbook fixes that.

This is not a Microsoft theory guide. This is an MSP implementation guide.

The Goal of This Playbook

By the end of this guide, you should have:

- A better understanding of Intune
- A defined baseline
- A rollout process
- A troubleshooting framework
- A deployment model that works across multiple tenants



What We're Going to Do Next

We're going to break down:

- What each policy type actually does
- When to use it
- When NOT to use it
- Where MSPs commonly double-configure standards
- How to avoid internal conflicts

Because once you understand the tools properly, then design becomes simple.

AUGMENTT

Chapter 1

The Intune Policy Mess (And Why It Happens Fast)

Let's be honest. Most MSPs don't fail at Intune because it's complicated; they fail because they treat it like Group Policy with a prettier UI. And that's where the mess begins.

The Pattern We See Over and Over

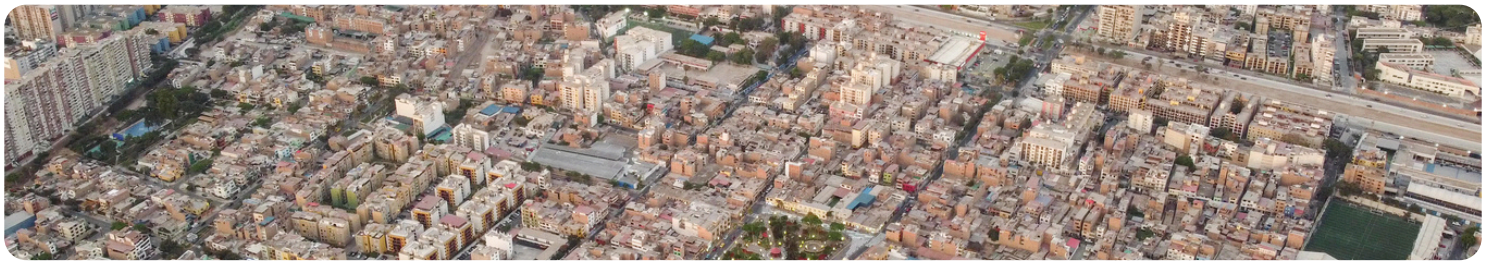
Here's how it usually goes:

1. Client buys Business Premium.
2. MSP "turns on Intune."
3. Someone deploys a set of Intune policies.
4. A few compliance policies get created.
5. A couple configuration profiles get layered in.
6. Everything gets assigned to "All Devices."

Three months later:

- ❌ Devices are conflicting.
- ❌ Policies overlap.
- ❌ Some settings live in Endpoint Security.
- ❌ Others live in Configuration Profiles.
- ❌ Nobody remembers why something was configured a certain way.
- ❌ A new client onboard breaks the "standard."

Welcome to policy sprawl.



AUGMENTT

Real-World MSP Scenario: How Chaos Actually Happens

Here's what this looks like in the real world.

Year 1	Year 2	Year 3
Tech #1 deploys a Microsoft Security Baseline.	Tech #2 adds endpoint security policies for Defender.	Tech #3 inherits the tenant.
A few custom configuration profiles get added.	No one checks whether those settings already exist elsewhere.	A new client requirement forces a change to BitLocker.
Everything is assigned to "All Devices."	A new compliance policy is layered in.	That change conflicts with two existing policies.
		Devices start reporting inconsistent encryption status.
		Nobody knows where the original setting lives.

Now troubleshooting becomes archaeology because there's:

- No documentation
- Multiple policy types configuring the same thing.
- No clean baseline.
- And no safe way to unwind it.

This is how MSP tenants become fragile. Not because Intune is bad, but because structure was never defined.

Why Intune Gets Messy Faster Than GPO Ever Did

With on-prem Group Policy, there was structure:

- OU hierarchy
- Inheritance
- Block inheritance
- Link order

Intune gives you freedom. Too much freedom. There's:

- Compliance policies
- Configuration profiles
- Endpoint security policies
- Security baselines
- Settings catalog
- Administrative templates
- Filters
- Scope tags
- App protection policies
- Remediations

All targeting the same devices. Without structure, it becomes policy roulette.

The Core MSP Problem

Enterprise IT builds for one tenant. MSPs build for 50. That changes everything.

You don't need "a working policy." You need a:

- Repeatable policy framework
- Deployment model that scales
- Structure that junior techs won't break
- Way to onboard new tenants cleanly

Intune doesn't give you that by default. You have to design it.

The Biggest Mistake: Deploying Before Designing

Most MSPs start clicking before they've defined:

- What is our baseline?
- What is optional?
- What is client-specific?
- What is device-specific?
- What is user-based vs device-based?
- What do we enforce vs just configure?

If you don't answer those first, every new policy becomes technical debt.

The Second Biggest Mistake: Mixing Policy Types Randomly

The same setting can live in multiple places. For example, Bitlocker settings can exist in:

- Endpoint Security
- Configuration Profiles
- Security Baselines

If you configure them in two places, you don't get "extra secure." You get conflicts. And troubleshooting conflicts in Intune is not fun.

The MSP Mindset Shift

Here's the shift that changes everything: Stop thinking in terms of policies. Start thinking in terms of layers.

1 — Core Security Baseline	*Applies to every managed device.*
2 — Platform-Specific Baseline	*Windows vs macOS vs iOS vs Android.*
3 — Role-Based Controls	*Executives, shared devices, frontline, kiosk.*
4 — Client-Specific Exceptions	*Temporary or contract-driven requirements.*

If you design around layers instead of policies, everything becomes manageable.

If You're Inheriting a Messy Tenant

Most MSPs aren't starting from zero. They're inheriting something...layered. If you're walking into policy sprawl, don't start deleting. Follow this process instead.

Step 1: Export Everything

You can't fix what you can't see. Export:

- Compliance policies
- Configuration profiles
- Endpoint Security policies
- Security baselines
- Assignments

Step 2: Identify Duplicate Ownership

- BitLocker settings in multiple places
- Defender configured in more than one policy type
- Firewall rules duplicated
- Password policies layered across profiles

Map which policy type "owns" each control. Pick one owner.

Step 3: Audit Assignments

Most chaos starts in assignments, not settings. Search for:

- "All Devices"
- "All Users"
- Overlapping nested groups
- Filter misuse

Step 4: Create a Pilot Safety Net

Before changing anything:

- Create a pilot group.
- Move test devices there.
- Make adjustments only against pilot first.

Never refactor production live.

Step 5: Consolidate Before Deleting

Do not:

- Delete Policies blindly.
- Turn off baselines without review.

Instead:

- Rebuild clean versions.
- Assign clean versions to pilot.
- Validate.
- Then remove legacy policies.

Unwinding safely is slower than building clean, but it prevents outages. Cleanup is surgery; not demolition.

Chapter 2 Understanding Intune Policy Types (Without Overcomplicating It)

If you don't understand what each policy type is responsible for, you'll duplicate settings and create conflicts.

Here's the simplified breakdown MSPs should use.

1. Compliance Policies: The Detective

Compliance policies don't configure devices; they evaluate them.

They answer one question: *What are the minimum requirements for devices to access company resources?*

Common checks:

- BitLocker enabled
- Password required
- OS version minimum
- Defender active
- Not jailbroken/rooted
- Antivirus enabled
- Firewall enabled

Compliance ties directly into Conditional Access. Think of compliance as your gatekeeper.

It only evaluates and grades. It does not:

❌ Configure settings
❌ Fix settings
❌ Deploy apps

2. Configuration Profiles: The Baseline Engine

This is where most of your real configuration lives. You can use configuration profiles to:

- Turn on BitLocker
- Configure Defender
- Set password policies
- Disable USB storage
- Configure Windows settings
- Push macOS restrictions

For MSPs, this is your core baseline layer. Use the Settings Catalog wherever possible for clarity and flexibility.

3. Endpoint Security Policies: Security-Focused Shortcuts

Endpoint Security policies overlap with configuration profiles. They focus on:

- Antivirus
- Firewall
- Disk encryption
- Attack surface reduction

They're easier to manage for security settings, but they can duplicate config profile settings.

Rule for MSPs: If you use Endpoint Security for something, don't configure it again in Settings Catalog. Pick one place and be consistent.

4. Security Baselines: Microsoft's Prebuilt Bundles

Baselines are large bundles of recommended settings. They're helpful for:

- Seeing recommended security posture
- Understanding what "good" looks like

They are dangerous, however, if deployed blindly, because they:

⚠️ Configure a lot at once
⚠️ May conflict with custom profiles
⚠️ May include settings you don't want

MSP advice: Use baselines as a reference, not as a blind deployment.

Important Warning About Security Baselines

Never deploy a Microsoft Security Baseline to production without reviewing every settings.

Baselines:

- Change over time.
- May include settings that break legacy apps.
- Can conflict with existing configuration profiles.
- May configure controls you didn't intend to enforce.

For MSP environments, a safer model is:

- Review baseline settings.
- Extract what aligns with your standard.
- Build those settings into your controlled baseline profiles.

Blind baseline deployment is one of the fastest ways to introduce policy conflicts. Use baselines as a reference material, not as a shortcut.

5. Application Policies

Apps are their own universe. There are many types to oversee:

- Win32 apps
- Microsoft Store apps
- Line-of-business apps
- macOS PKG
- iOS VPP
- Android Managed Play

Apps are where MSPs spend most operational time. We'll spend more time on this in chapter 7.

6. Remediations (Proactive Remediation Scripts)

Note: You will need one of the following licenses to access Remediations:

- *Windows Enterprise E3 or E5 (included in Microsoft 365 F3, E3, or E5)*
- *Windows Education A3 or A5 (included in Microsoft 365 A3 or A5)*
- *Windows Virtual Desktop Access (VDA) per user*

These are powerful. They:

✅ Detect an issue
✅ Run a script to fix it

Perfect for:

✅ Registry corrections
✅ Drift control
✅ Enforcing niche standards

Think of remediations as your safety net.

Chapter 3 Groups, Assignments & Targeting (The Part That Breaks Everything)

If your targeting structure is wrong, nothing else matters.

The First Rule: Stop Assigning to "All Devices"

"All Devices" should almost never be used. Why? Because it:

- ❌ Removes control
- ❌ Makes pilots impossible
- ❌ Breaks exception handling
- ❌ Makes troubleshooting harder

Device vs User Targeting

This decision impacts everything. Most MSPs default to device groups because it feels more controlled. In modern Entra ID + Autopilot environments, user-based assignment is usually the cleaner default.

When you assign to users:

- ✅ Policies follow the person
- ✅ Compliance follows the person
- ✅ Apps follow the person
- ✅ OOBE / Autopilot is smoother

As soon as a licensed user signs in, policies begin applying.

With device targeting, the device must:

- Enroll
- Register
- Land in the correct group

Before policies apply. That delay can create drift, especially in busy MSP environments.

For most MSP baselines:

- Compliance → User groups
- Core configuration → User groups
- Apps → Mostly user groups

Use device groups for:

- Kiosks
- Loaners
- Shared for fixed-purpose machines

Simple rule:

If it should follow the person, *assign to users.*

If it must stay tied to hardware, *assign to devices.*

Get this right early. It either creates scale...or tickets.

The MSP Group Model (Recommended)

Create a clean top-level structure first.

1. Core Platform Groups

- All Managed Windows
- All Managed macOS
- All Managed iOS
- All Managed Android

These are your baseline anchors.

Under those, nest logical sub-groups for flexibility. For example:

All Managed Windows
- *Engineering*
- *Finance*
- *Sales*

You can even go deeper when needed:

Engineering
- *Development*
- *Testing*

All nested groups inherit configuration policies from the parent.

That gives you structure **and** controlled forking when departments need different configurations.

2. Pilot Groups

- Windows Pilot
- macOS Pilot

Keep these separate from your production structure. Everything new should hit pilot first.

3. Role-Based Groups

- Executives
- Shared Devices
- Kiosk

Use these for targeted controls that cut across departments.

4. Exception Groups

- Temporary overrides

Keep them isolated and documented. Exceptions should be intentional, not accidental.

Important caveat: Use nesting for configuration policies.

Avoid nesting for application deployments. App assignments don't always trickle down reliably through nested groups, and that can create inconsistent installs.

Design the structure once. Keep it clean. Replicate it across tenants.

Structure is what allows you to scale without rebuilding every client from scratch.

Document the Why, Not Just the What

Policies without documentation don't scale. It's not enough to know what a policy does. You need to know why it exists.

For every exception group, document:

- The business reason
- The approval source
- The intended expiration date
- The policy owner

Temporary exceptions that aren't documented become permanent drift.

For baseline changes:

- Version them.
- Log what changed.
- Record why it changed

Do not rely on naming conventions alone as documentation.

Over time, staff will turn over, tenants will grow, and memory will fade. Structure without documentation eventually collapses.

Use Filters Carefully

Filters are powerful. They allow:

- OS version targeting
- Device model targeting
- Ownership targeting

But overusing filters adds complexity.

MSP rule: Groups first. Filters second.

Chapter 4

Compliance Policies: Your Enforcement Engine

Compliance should be simple. If it's complicated, you're doing too much.

Recommended MSP Default Compliance (Windows)

- BitLocker required
- Password required
- Minimum OS version enforced
- Secure Boot enabled
- Antivirus enabled
- Firewall enabled
- Defender enabled (if not using your own 3rd party solution)

That's it for most SMB clients. You don't need 40 checks.

Tie It to Conditional Access

Compliance without Conditional Access is just reporting. Conditional access should:

- Block non-compliant devices
- Block unmanaged devices (if client is ready)
- Require security registration from managed device

Roll this out carefully. Start in report-only mode.

Conditional Access Rollout Strategy (Do Not Skip This)

Flipping on Conditional Access without preparation is how MSPs create emergency tickets. Follow this rollout model:

Phase 1
Report-Only Mode

Enable policies in report-only. Monitor for 2-4 weeks.

Watch for:
- Legacy device failures
- Shared devices
- Line-of-business app issues
- Executives using unmanaged personal devices

Phase 2
Limited Enforcement

Enforce against:
- IT staff
- Pilot users
- Friendly departments

Measure impact before widening scope.

Phase 3
Broad Enforcement

Do not flip enforcement globally in one day; roll out gradually.

Communicate with the client before enforcement, because security enforcement without stakeholder awareness creates friction. Security with communication builds trust.

What Not to Do

- ✗ Don't create different compliance policies per department.
- ✗ Don't enforce overly strict OS version requirements on day one.

Chapter 5

Configuration Profiles: Building a Clean Baseline

This is your foundation.

Windows Baseline (MSP Recommended Structure)

Create separate profiles for:

1. Device Restrictions
2. Password Policy
3. BitLocker (if not using Endpoint Security)
4. Defender configuration
5. Update Rings
6. Settings Catalog (if needed)

Keep them modular. Don't create one 300-setting monster profile.

macOS Baseline

- FireVault enabled
- Password policy enforced
- Firewall enabled
- System extensios approved
- Update settings defined

Test carefully. macOS behaves differently than Windows.

iOS / Android

Focus on:

- Passcode enforcement
- Encryption
- OS version
- App control
- Corporate data separation

Keep mobile lighter than desktop.

Naming Convention

If your naming is messy, your tenant will be messy.

Recommended format:

```
[Platform] – [Layer] – [Function] – v1
```

Example:

WIN — Baseline — Defender — v1

Consistency beats creativity.

Chapter 6

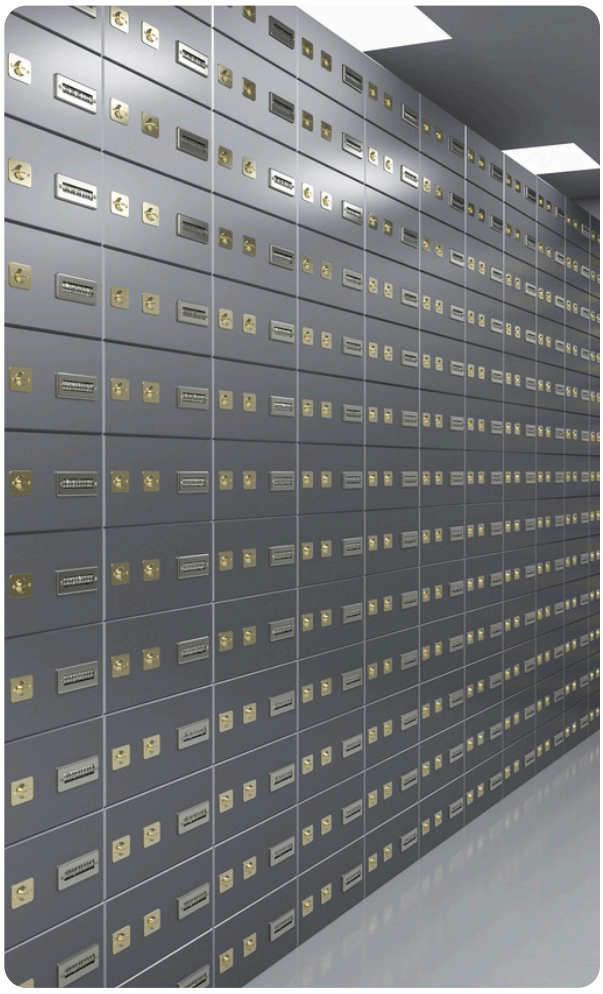
Endpoint Security & Defender (Where MSPs Get Confused)

This is where duplication happens.

You can configure Defender in:

- Settings Catalog
- Endpoint Security
- Security Baselines

Pick one.



Recommended Approach

Use Endpoint Security for:

- Antivirus
- Firewall
- Disk encryption

Use Settings Catalog for:

- Non-security OS controls

Keep security centralized.

Common Conflict Scenario

BitLocker configured in:

- Endpoint Security AND
- Configuration Profile

Result:

- Reporting inconsistencies
- Confusing error states

Fix: *Consolidate.*

Licensing Impacts Architecture

Not every client has the same Microsoft license tier. Your baseline design must account for that. For example:

- Remediations require specific Windows Enterprise licensing.
- Defender feature depth varies by SKU.
- Certain reporting capabilities require higher tiers.

If you standardize across tenants, define your minimum supported license level. Then:

- Build your architecture around that.
- Document deviations for lower-tier clients.
- Avoid designing features you can't deploy consistently.

Licensing isn't just billing; it shapes your security architecture.

Chapter 7

Application Deployment: The MSP Reality

Apps consume more time than policies.

Win32 Apps

These are the most powerful and flexible, but also the most work. Use for:

- Complex installs
- Silent installs
- Scripted installs

Always test detection rules carefully.
Bad detection = constant reinstall loops.

Required vs Available

Required	Available
Installs automatically	User installs from Company Portal
Good for security tools	Good for optional tools

MSP tip: Don't auto-install everything.

Wingnet Reality

Wingnet is promising, but:

❌ Not every app behaves cleanly
❌ Version control can be inconsistent

Test before standardizing

App Deployment Best Practice

1. Package
2. Test on pilot group
3. Validate detection
4. Roll to production

No shortcuts.

Chapter 8

Rollout Strategy: Pilot > Phased > Production > Maintain

This is where most MSPs rush. Don't.

Phase 1: Build in Isolation

Create all policies unassigned first. Review for overlap.

Phase 2: Pilot

Assign to:

- IT devices
- Friendly client users

Monitor for:

- Conflicts
- Performance issues
- App failures

Phase 3: Gradual Expansion

Expand in waves: 10%, 25%, 50% and 100%.

Track support tickets. If tickets spike, pause.

Phase 4: Maintain & Evolve

Production is not the finish line. It's the starting point of ongoing management.

Once fully deployed, build recurring operational checks for:

- Compliance percentage trends
- Failed configuration policies
- Inactive devices
- Application version updates
- New Microsoft security settings and features

Security baselines change. Windows builds change. Apps update. If you don't review policies regularly, drift creeps in. Set a recurring review cadence:

- **Monthly:** Compliance and failed policy review
- **Quarterly:** Baseline and configuration review
- **As-needed:** App version packaging and deployment updates

Deployment gets you live, while maintenance keeps you secure.

No Production Changes Without Pilot

Once you reach production, the temptation begins:

- Microsoft releases a new setting.
- A security blog recommends a change.
- An engineer wants to "just turn something on."

Don't make direct production changes. Even small adjustments should:

1. Hit pilot first.
2. Be validated.
3. Be observed for support impact.
4. Then move to production.

Drift often re-enters environments through uncontrolled improvement. Mature MSPs treat changes as deployments, not tweaks.

Chapter 9

Monitoring, Reporting & Ongoing Operations

Intune is not “set and forget.” Without structure, drift is guaranteed.

What to Monitor Weekly

- Compliance rate
- Failed configurations
- Failed app installs
- Devices not checking in

When Policies “Don’t Apply”

Check:

1. Is device in correct group?
2. Conflicting setting?
3. License assigned?
4. Device sync recently?

Troubleshooting is systematic. Not guesswork.

Drift Control

Use:

- Remediations
- Clear baselines
- Standardized structure across tenants

Compliance Alerting

Dashboards are passive. MSPs need proactive visibility.

If compliance drops, you shouldn't find out from the client.

Configure alerts for:

- Devices becoming non-compliant
- Sudden compliance drops
- Devices not checking in

Send notifications to a monitored service desk or security mailbox — not an individual tech.

If you're enforcing Conditional Access, you need to know immediately when devices start failing.

Enforcement without alerting is blind enforcement.

Chapter 10

The MSP Intune Baseline Checklist

Core Setup

- ☐ Group model defined
- ☐ Naming convention defined
- ☐ Core device groups created
- ☐ Pilot groups created

Baseline

- ☐ Windows baseline built
- ☐ macOS baseline built
- ☐ Mobile baseline built

Security

- ☐ Endpoint Security policies configured
- ☐ Compliance tied to Conditional Access

Applications

- ☐ Standard app stack packaged
- ☐ Detection rules validated
- ☐ Pilot tested

Rollout

- ☐ Pilot phase complete
- ☐ Gradual rollout executed
- ☐ Reporting reviewed

Final Thoughts

Clients don't see your policy design. They feel the outcome:

- Fewer login issues.
- Predictable device behavior.
- Clean onboarding.
- Stable security enforcement.
- Fewer surprise lockouts.

Structured Intune isn't just technical hygiene. It's operational leverage.

- It reduces tickets.
- It reduces engineer burnout.
- It improves client confidence.
- And it allows you to scale without rebuilding every tenant from scratch.

That's the difference between managing Intune and mastering it as an MSP.

AUGMENTT

See how Augmentt helps MSPs operationalize Microsoft security.

augmentt.com