Intro to Monitoring

For AV Professionals

> whoami 🤘

- Frederick Loucks, CTS-D
- By day: TPM @ Zoom
- By night: consultant & educator
- My AV Origin Story
  - Music & Theatre background
  - 2x SCN Top 50 integrators
  - 2x Fortune 500 enterprises
- What do I get excited about?
  - Books: fantasy & sci-fi
  - Music: power & progressive metal

Introductions

Please share your name, role, where you are joining from, and one thing you hope to learn today.

Monitoring

Agenda

Welcome

- What is monitoring and why should we care about it?
- Why is monitoring AV systems so hard?
- How do we design systems that are easier to monitor?
- What should we consider when selecting a monitoring platform?
- What are the steps to get started?
- Where does AV/IT convergence fit in?

Monitoring: What & Why

What is monitoring?

Mainstream definitions

Monitoring: What & Why

Splunk:

> Any processes and tools you use to determine if your organization's IT equipment and digital services are working properly. Monitoring helps to detect and help resolve problems — all sorts of problems.[1]

Google:

> Collecting, processing, aggregating, and displaying real-time quantitative data about a system, such as query counts and types, error counts and types, processing times, and server lifetimes.[2]

1. Splunk ↩
2. Google SRE Handbook ↩

Fred’s monitoring razor

Monitoring is…

Knowing the health of your systems at any given point-in-time.

Going deeper: 4 key terms

Monitoring: What & Why

- **Telemetry** (*noun*):
  - The data we collect from our systems.
- **Monitor** (*verb*):
  - The act of collecting & processing realtime telemetry for some business purpose.
- **Observable** (*adjective*):
  - The degree to which we can understand the state of a system by monitoring its telemetry.
- **Operator** (*noun*):
  - The person who is responsible for monitoring & managing the system.
- The more complete the telemetry is and the more effective the monitoring is, the more observable the system is, and the better supported the operator is.

Why should we monitor?

Reasons to monitor

Monitoring: What & Why

- Know the health of your systems at any given point-in-time
- Improve system reliability and uptime
- Shift from reactive to proactive
- Improve user & operator experience
- Decrease costs through efficiency
- Provide a wealth of data for analysis and improvement
- Build job security and growth opportunities
- Satisfy your own technical curiosity and desire to learn

Maybe the question isn’t why should we monitor…

…but why don’t we monitor?

Why don’t we monitor?

Common reasons for not monitoring

- Don’t know where to start
- Lack of tools / platforms
- No budget / resources / time
- Can’t make financial sense of it
- It’s just too difficult / doesn’t seem worth it
- Nobody has asked for it / no perceived need

Monitoring AV is hard

Why is it so hard to monitor AV systems?

Why is monitoring AV hard?

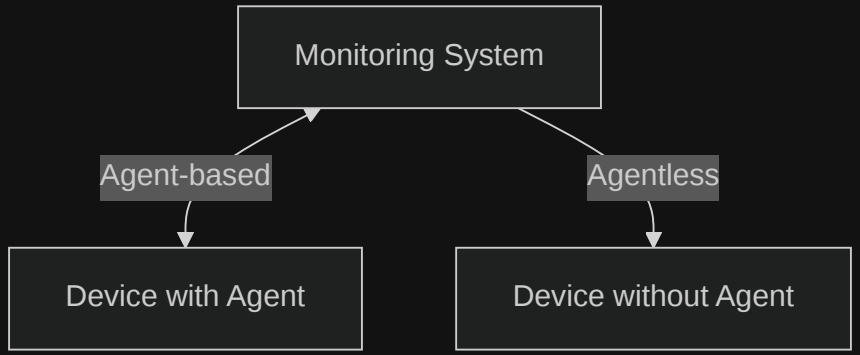
Monitoring AV is hard

- Hardware & firmware means agentless
- Limited API standards or documentation
- Integrated systems have complex health models
- Insufficient commercial pressure
- Conflicts of interest

Hardware & firmware means agentless

Monitoring AV is hard

- Typically monitoring is agent-based
  - Fetch locally & push
- Firmware-based hardware appliances can’t run agents
- Can only monitor what the device provides
  - Sometimes push, usually poll
- Increases complexity and integration requirements
  - A new integration per device model / API



Limited API standards or documentation

Monitoring AV is hard

- While control APIs tend to be pretty good, management APIs are not
- No consistency in port, protocol, or data format
- Wildly varying levels of completeness
- No standardization across manufacturers, products, or even FW version
- On top of that, poor documentation

Integrated systems have complex health models

Monitoring AV is hard

- Measuring system health is the #1 most important thing in monitoring
- Health models codify the impact of potential failures in a system (🔴🟡🟢)
- AV systems are complex with many component interactions
  - More complexity, more potential failure points
- Each unique combination of gear requires a new health model
  - Large volume of bespoke systems makes this unsustainable
- Standards required for health modeling, otherwise unfeasible

Insufficient commercial pressure

Monitoring AV is hard

- Manufacturers hold a lot of the cards here
- Insufficient commercial pressure to provide open, documented, stable APIs
- Not enough customers are demanding it / voting with their wallets

Conflicts of interest

Monitoring AV is hard

- Many manufacturers want to sell you their own management tools
- Providing open monitoring APIs would compete with that
- Often not incentivized to make it easier for you to monitor your systems if it means you won't buy their tools

There’s a lot of that we can’t control…

But there are some things we can!

So what can we do?

Monitoring AV is hard

- Spec observable products and design observable systems
  - Subset of building systems that are manageable at scale
- Choose monitoring platforms that are flexible and support our needs
- Have a plan, start small, and iterate

Designing observable systems

Observable systems require observable components

Designing observable systems

- Devices
  - Six-Factor Device
- Applications
  - Local / per-system
  - Cloud / multi-system
- Network Services
- Environments
- System Health

Devices

Designing observable systems - Devices

Fred's "Six-Factor Device"

1. Open API
2. Standardized
3. Hardened
4. Observable
5. Remote-First
6. Declarative

Open API

Designing observable systems - Six-Factor Device

- Included with device purchase
- Local to the device, not proxied through a cloud service
- Well-documented and easy to reference
- Comprehensive, covering all device features
  - Monitoring, control, configuration, etc...more on this later
- Stable and versioned

Standardized

Designing observable systems - Six-Factor Device

- Leverages standard technologies
  - API: HTTP, REST/RPC, JSON, GraphQL, MQTT, etc.
  - CLI: SSH
  - Configuration: JSON, YAML
  - Monitoring: SNMP, Syslog, HTTP, WebSockets, MQTT etc.
- Avoids proprietary & bespoke solutions
- Widely supported by standard programming languages and monitoring / management tools

Hardened

Designing observable systems - Six-Factor Device

- Implements security best-practices by default
- Encryption in transit
  - HTTPS for API and web interfaces
  - CLI access via SSH
- Reset admin password on first boot
- Allow disabling of insecure or unused services
  - Web interface, console, etc.
- Regularly updated firmware, managed via API

Observable

Designing observable systems - Six-Factor Device

- Surfaces all state, performance, and event data via the API
  - State data: current settings, status, etc.
  - Performance data: CPU, memory, network, etc.
  - Event data: logs, alerts, etc.
- Available via push or poll methods
  - Poll: HTTP, SNMP, SSH, etc.
  - Push: WebSockets, SNMP Traps, MQTT, etc.
- Data is human-readable and machine-readable
  - JSON, XML, etc.
  - Timestamped and tagged with metadata

Remote-First

Designing observable systems - Six-Factor Device

- All management and control tasks are possible via the API
  - Control: power, volume, input, etc.
  - Management: configuration, firmware updates, etc.
  - Monitoring: status, performance, events, etc.
- No desktop software or local network access required
- No physical access to the device required

Declarative

Designing observable systems - Six-Factor Device

- Uses declarative, human-readable configuration via the API
- Configuration is stored in a human-readable format
  - JSON, YAML, etc.
- Configuration is versioned and can be backed up
- Configuration can be fetched or applied via the API
- Configuration can be checked into source control

Recap: Manageable and Scalable Systems

Designing observable systems - Devices

Fred's "Six-Factor Device"

1. **Open API** - Implements an open, comprehensive, and well-documented API
2. **Standardized** - Leverages standard technologies, avoids proprietary solutions
3. **Hardened** - Implements security best-practices by default
4. **Observable** - Surfaces all state, performance, and event data via the API
5. **Remote-First** - All management and control tasks are possible via the API
6. **Declarative** - Uses declarative, human-readable configuration via the API

Observable devices != observable systems

There’s more to it than just the devices

Applications - Local / Per-System

Designing observable systems

- Control systems, DSP, etc.
- Capture business logic and user interactions
- Can have robust APIs
- Often developed by systems integrators
- Frequently use proprietary technology
- Can serve as critical middleware

Applications - Cloud / Multi-System

Designing observable systems

- VC, UCC, CDMP, Calendar, etc.
- Often cloud-based
- Often have robust APIs
- Can be difficult to observe due to security practices
- May require additional licensing or permissions
- Considered middleware for our purposes

Network Services

Designing observable systems

- LAN, WAN, internet, DNS, NTP, etc.
- Important for streaming media
- Can be a source of performance issues
- Owned and monitored by IT, but impact AV systems
- Leverages standard protocols like SNMP, NetFlow, TCP checks etc.

Environment

Designing observable systems

- Sensors, thermostats, etc.
- Impact user experience
- Provide valuable usage data
- Often overlooked
- Privacy concerns

System Health

Designing observable systems

- Sometimes called "health modeling" or "service monitoring"
  - The same "service" that is found in "Service Level Agreement"
- Components all come together to determine the health of the system
  - Each component can trigger alerts based on its own health
  - Each trigger impacts overall system health in well-defined ways
- The "uptime" of the system is a function of the health of its components
  - This metric is called the "Service Level Indicator" (SLI)
  - The target uptime is the "Service Level Objective" (SLO)
  - The contract between the provider and the customer is the "Service Level Agreement" (SLA)
- An uptime SLA is impossible without a health model

System Health - Example

Designing observable systems

Component	Problem	Severity
Display	Device unreachable	Average
Display	Poor network connectivity	Average
Display	Display powered off while in use	Average
Display	Display on the wrong input while in use	Average
Display	Brightness outside of target	Warning
Display	Display volume outside of target	Warning
Video Bar	Device unreachable	Average
Video Bar	Poor network connectivity	Average

Recap: Observable systems require observable components

Designing observable systems

- Devices (AV Gear)
  - Six-Factor Device
- Applications
  - Local / per-system (DSP, control systems)
  - Cloud / multi-system (CDMP, UCC, VC, Booking)
- Network Services (LAN, WAN, internet, DNS, NTP)
- Environments (sensors, occupancy, temperature)
- System Health (brings it all together, measures health with SLI/SLO/SLA)

Monitoring Platforms

What is a monitoring platform?

Definition

Monitoring platforms

- A monitoring platform is a software system that
- collects, processes, and displays telemetry data from systems
- in order to help operators understand the state of the system,
- detect and resolve issues before they become problems,
- and optimize system performance.

Reminder: 4 key terms

Monitoring platforms

- **Telemetry** (*noun*):
  - The data we collect from our systems.
- **Monitor** (*verb*):
  - The act of collecting & processing realtime telemetry (data) for some business purpose.
- **Observable** (*adjective*):
  - The degree to which we can understand the state of a system by monitoring its telemetry.
- **Operator** (*noun*):
  - The person who is responsible for monitoring & managing the system.
- The more complete the telemetry is and the more effective the monitoring is, the more observable the system is, and the better prepared the operator is.

Types of Monitoring Platforms

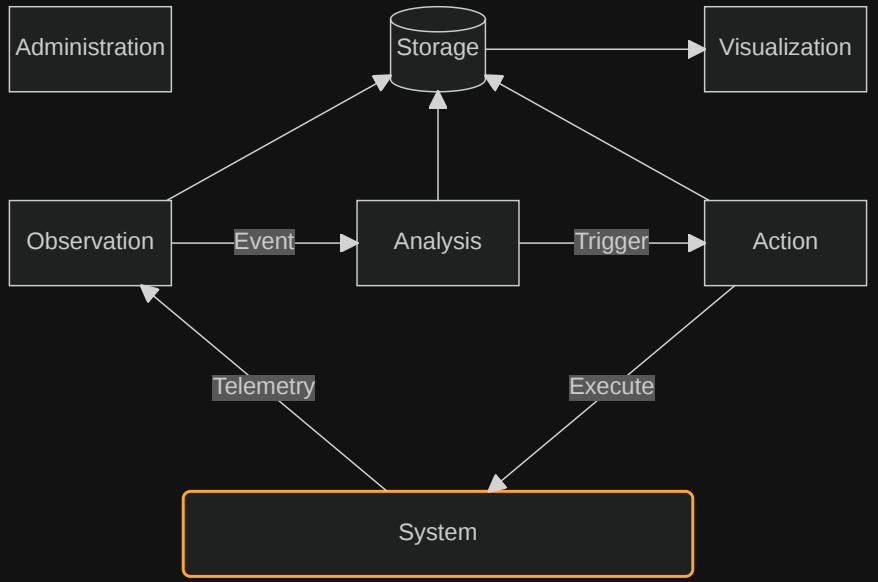
Monitoring platforms

- **Generic (infra, network, availability)**
  - Flexible, general purpose, customizable
  - Sometimes open source, sometimes commercial
- **Cloud Device Management Platform** (CDMP)
  - Native integration, OEM-specific
  - Useful as middleware but not for a single pane
- **AV Monitoring**
  - Typically built on top of control systems
- **Application Performance Monitoring** (APM)
  - Focuses on appllication development and performance
  - Often requires data to be pushed to the monitoring platform
- **Security Monitoring** (SIEM, IDS, etc.)
  - Focuses on security threats and incidents

6 Core Functions of a Monitoring System

Monitoring platforms

- **Observation**: Collect, detect
- **Analysis**: Correlate, calculate, filter
- **Action**: Notify, respond
- **Storage**: Retain, serve
- **Visualization**: Dashboard, report, browse
- **Administration**: Configure, maintain, secure



* Inspired by

Key areas of concern for a monitoring platform

Monitoring platforms

1. Agentless monitoring
2. Single pane of glass
3. Flexible data collection
4. Scalability
5. Customizable / extensible
6. Remote access
7. Tunable alerts & actions
8. Granular user permissions
9. Pricing that works for AV
10. Advanced visualizations
11. Service monitoring with health models
12. Community support
13. Commercial support
14. Extensive documentation
15. Well established release cycle

Agentless monitoring

Monitoring platforms - Key areas of concern

- Not Mac, Windows, Linux, etc - No agents
- Leverages collectors to pull data from devices
- Flexible collector hosting (Bare metal, Virtual machine, Container)
- Buffers data when internet drops
- Collector high-availability
- Extensible with custom scripts, binaries, etc.

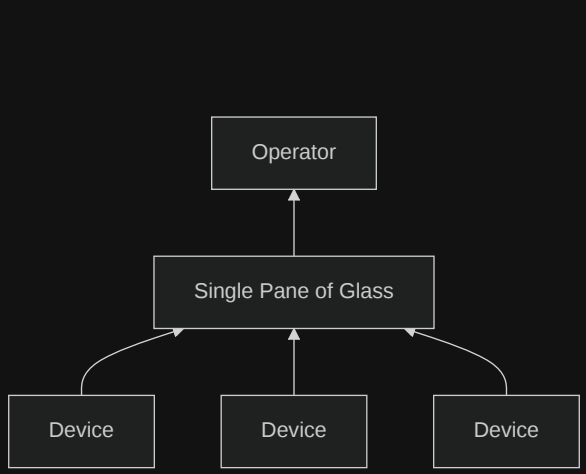
Single pane of glass

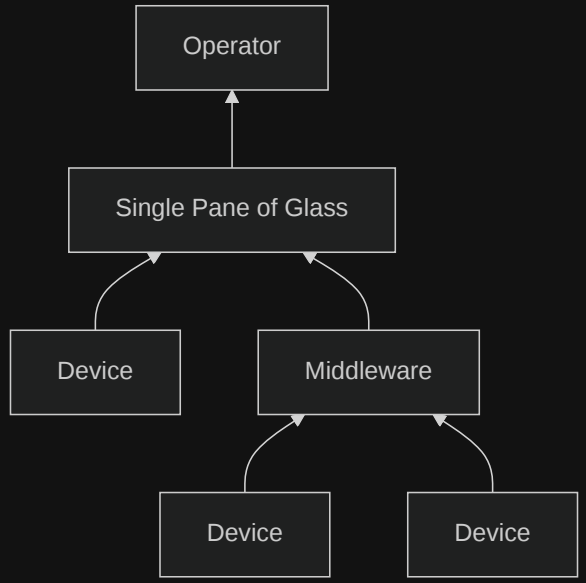
Monitoring platforms - Key areas of concern

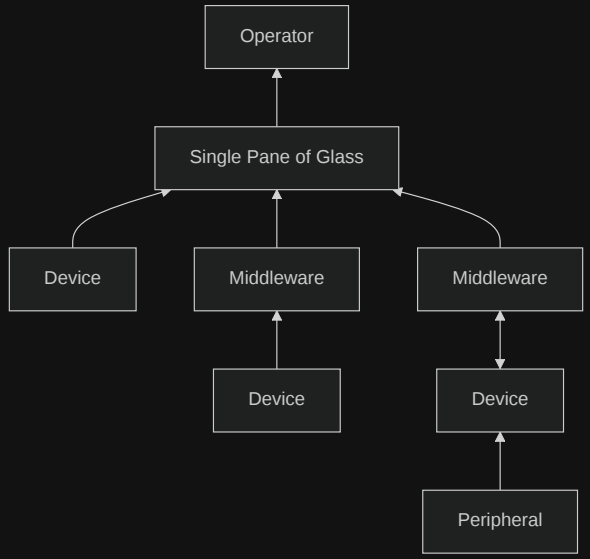
- All devices, systems, services in one place
- Centralized processing of data and alerts
- Manage dependencies between devices & services
- Single database of all assets
- One place to configure alerts, actions, etc.
- Single pane does not necessarily mean single tool

Single Pane of Glass != Single Tool

Monitoring platforms - Key areas of concern







Flexible data collection

Monitoring platforms - Key areas of concern

- Due to variability in device APIs, maximum flexibility is required
- Protocol support should include:
  - ICMP, SNMP, UDP & TCP simple checks, Console checks (SSH, Telnet, Raw TCP), HTTP/s, Web Driving, and more
  - Custom script + arbitrary binary
- Push metrics from devices to platform
  - Middleware, control systems, etc.
- Flexible parsing & filtering capabilities
  - Regular expressions, JSONPath, XPath, etc.
- Dynamic creation of metrics and devices from scripts, queries, scans

Scalability

Monitoring platforms - Key areas of concern

- Device proliferation is a significant issue
- Scalability is a key concern
- Must have templates with inheritance
  - Metrics, triggers, graphs, dashboards, discovery rules, etc.
  - Configure in one place, apply to thousands of devices
- API-driven configuration when required
- Distributed monitoring capabilities
- Cloud-native or cloud-ready via containers

Customizable / extensible

Monitoring platforms - Key areas of concern

- Open source vs. commercial
- Open APIs
- Plugin architecture
  - SNMP MIBs, custom scripts, etc.
  - Dashboard widgets
  - Custom pages
- Community contributions

Remote access

Monitoring platforms - Key areas of concern

- Remote access to devices & services
- Tunneling & proxying vs direct
- Security
- Interfaces
  - Web pages
  - SSH, Telnet, Raw TCP
  - VNC, RDP
- Port forwarding
  - Proxy to host
  - Limited time & latching
- VPN
  - On-demand & inside-out
  - Supports desktop tools

Tunable alerts & actions

Monitoring platforms - Key areas of concern

- Alerts
  - Alert fatigue is real
  - Coorelation with similar alerts, parent, child, sibling
  - Acknowledgement by ops staff
  - Suppression when needed
  - Escalation policies including scripted actions
- Notifications via
  - Email, chat, SMS
  - Webhook
  - 3rd party integrations
- Actions
  - Run scripts on demand and on event
  - Arbitrary binaries for max flexibility
  - Restart services, reboot devices, etc
  - Set / get configurations
  - Perform local server management
- Alert + Action = Automation / Self-Healing

Granular user permissions

Monitoring platforms - Key areas of concern

- Strong admin controls
- Role-based access control with flexible scoping
- Full CRUD controls for all objects
- Management of users by group
- Built-in and custom roles
- Permissions for devices, groups, templates, etc.
- Audit logs
- SSO, MFA, SCIM

Pricing that works for AV

Monitoring platforms - Key areas of concern

- Depends on how you will fund it
- Open source vs. commercial
- Per device, per room, per user, per metric, per feature, per data consumption
- Edge component pricing
- Add-ons for advanced features
- Support costs
- 3rd party development needs

Advanced visualizations

Monitoring platforms - Key areas of concern

- Dashboards
  - Widget & canvas
  - Line, bar, pie, gauge, map, etc.
  - Dynamic or static
  - Drill-down
  - BYO widgets
- Maps
  - Device, network, service
  - Customizable
- Reports
  - Customizable
  - Scheduled
- Custom pages
  - AV-specific views
  - Control system / CDMP integration

Service monitoring with health models

Monitoring platforms - Key areas of concern

- Each system is a service
- Define health model for each & control bubble-up of issues
- Service level agreement monitoring & reporting
- Flexible service trees (service heirarchy)
- Service actions for automated responses

Community support

Monitoring platforms - Key areas of concern

- Active community
- Forums, chat, etc.
- "Google-ability" of issues
- Shared library of scripts, templates, etc.
- Easier to hire for than niche tools

Commercial support

Monitoring platforms - Key areas of concern

- Paid support for when things go wrong
- Many open source companies make money from enterprise support
- Provided direct or through partners
- SLAs (response time & resolution time)
- Professional services to help get started

Extensive documentation

Monitoring platforms - Key areas of concern

- Critical for any serious monitoring platform
- Google "$vendor docs" before you buy
- Solid documentation is worth hundreds of hours
- Must be up-to-date and searchable
- Should include examples and use cases

Well established release cycle

Monitoring platforms - Key areas of concern

- Regular updates on a fixed cycle
- Clear release notes with changelog
- Published roadmap for future features
- Ability to submit feature requests

Recap: Key areas of concern for a monitoring platform

Monitoring platforms

1. Agentless monitoring
2. Single pane of glass
3. Flexible data collection
4. Scalability
5. Customizable / extensible
6. Remote access
7. Tunable alerts & actions
8. Granular user permissions
9. Pricing that works for AV
10. Advanced visualizations
11. Service monitoring with health models
12. Community support
13. Commercial support
14. Extensive documentation
15. Well established release cycle

Getting Started

For Manufacturers

Where do we go from here

Getting started for manufacturers

1. Assemble the team
2. Set clear goals
3. Define your product monitoring strategy
4. Choose a monitoring architecture
5. Start small & iterate

Assemble the team

Getting started for manufacturers

- End users/customers
- Software development / engineering
- Product management
- IT / Infrastructure
- Support & service teams
- Sales & marketing
- Channel partners
- Security & compliance

Assess skills on staff

Getting started for manufacturers - Assemble the team

- Evaluate if your hardware teams have the necessary software expertise
- Consider creating a dedicated monitoring & management product team
- Leverage partnerships with software companies for monitoring platforms
- Invest in upskilling engineering teams on cloud, APIs, and data analytics
- Consider acquiring companies with monitoring expertise
- Hire team members with experience from IT monitoring sectors
- Evaluate if your organization has the cybersecurity expertise for secure monitoring

Set clear goals

Getting started for manufacturers

- Goals define what value monitoring features will deliver for products
- They inform architecture and investment decisions
- Be SMART
  - Specific, Measurable, Achievable, Relevant, Time-bound
- One person should ultimately be responsible for each goal
- Balance product, business, and customer goals
- May focus on new revenue generation, increased consumption of existing products, IoT data collection & analysis, customer satisfaction / loyalty, etc.

Define your product monitoring strategy

Getting started for manufacturers

- Build, buy, or partner for monitoring capabilities
- Determine what data your products should expose and how
- Design secure, scalable, standardized APIs for third-party integration
- Establish data management and privacy policies
- Consider monetization opportunities
- Plan for lifecycle management of monitoring features
- Create clear documentation and developer resources

Monetization models

Getting started for manufacturers

- Free Offer
  - Open APIs for data collection of events, logs, and metrics
  - Creates competitive advantage in marketplace, drives customer loyalty and reduces support costs
- Premium Offer
  - Advanced monitoring features (ie Faster metric / event intervals, retention, key integrations)
  - Focus on enterprise use-cases
- SaaS Offer
  - Fully managed monitoring platform
  - Consumption-based pricing models (data volume, seats)
  - Tenant license, beware per-device price
  - Integration with existing cloud services
  - API-gateway
  - Beware: we don’t need more of these

Choose a monitoring architecture

Getting started for manufacturers

- Balance device-level vs. system-level monitoring
- Consider embedded monitoring vs. external agents
- Evaluate push vs. pull data collection models
- Define storage strategy for telemetry data
- Plan for scale - from single devices to enterprise deployments
- Design for multi-tenancy and role-based access
- Ensure security is built into architecture from the start
- Consider backward and forward compatibility

Technical considerations for products

Getting started for manufacturers

- Design monitoring capabilities as a core feature, not an afterthought
- Implement standardized APIs across product lines
- Build fault-tolerant monitoring that works even when products malfunction
- Include local buffering for telemetry when network connectivity is lost
- Design a secure monitoring infrastructure resistant to attacks
- Create efficient telemetry that minimizes bandwidth and processing overhead
- Implement semantic versioning for all monitoring interfaces
- Test monitoring at scale simulating real-world deployments

Start small & iterate

Getting started for manufacturers

- Begin with a pilot on a single product line
- Partner with loyal customers and integrators for feedback
- Establish structured feedback loops with internal and external users
- Create a clear roadmap with iterative improvements
- Capture telemetry to measure monitoring feature adoption
- Allow for experimentation with different approaches
- Plan for regular updates based on real-world usage data
- Don’t assume you’ll get it right the first time

Review: Getting Started for Manufacturers

Getting started for manufacturers

1. Assemble the team - cross-functional expertise spanning hardware, software, and business
2. Set clear goals - specific, measurable outcomes for products, business, and customers
3. Define your product monitoring strategy - data collection, architecture, APIs, monetization
4. Choose a monitoring architecture - secure, scalable, and appropriate for your products
5. Start small & iterate - pilot, gather feedback, continuously improve

AV/IT Convergence

Bridging Two Worlds for Enhanced Collaboration

Overview

- Introduction to AV/IT Convergence
- Key Differences Between Domains
- The Blurred Line: Monitoring & Management
- Integration Challenges
- Key Takeaways

Key Differences

Core Focus

Key Differences

AV

- Real-time audio and video quality
- Low-latency performance requirements
- User experience prioritization
- Hardware-centric solutions
- Signal integrity and distribution
- Physical space considerations

IT

- Data integrity and protection
- Network reliability and scalability
- Security and compliance
- Software-driven systems
- Resource optimization
- Standardization
- Process-oriented approach

Historical Approach to Problems

Key Differences

AV Approach

- Project-based implementation
- Custom solutions for unique requirements
- Manual configuration and maint.
- On-site, hands-on troubleshooting
- Specialized / vendor-specific tools and protocols
- Success measured by user feelings

IT Approach

- Service-based delivery
- Standardized solutions at scale
- Automated config and management
- Remote monitoring and troubleshooting
- Universal tools and protocols
- Success measured by KPIs

Both AV & IT are evolving.

Transformational Trends

Evolution

AV Evolution

- Transition to IP-based AV systems
- Software-defined AV systems
- Shift from hardware to services
- API-driven integration / interop
- Cloud-connected devices / IoT
- Data analytics for system optimization

IT Evolution

- Increased focus on collaboration tools
- Support for real-time applications
- Edge computing for local processing
- User experience prioritization
- Multi-media content management
- Low-latency network optimization

The line between AV & IT technology is blurring.

Technology Convergence

Blurred lines

- IP-based AV systems integrate with IT tools
- Common data collection & storage methods
- Key metrics:
  - Device & system health and uptime
  - System utilization
  - Underlying network performance
  - User experience metrics
  - UCC platform health
- Converged alerting systems

What AV Can Learn From IT

Blurred lines

- **Standardization** of processes and practices
  - Reduces complexity and improves scalability
  - Enables consistent user experiences across spaces
  - Facilitates training and support
- **Automation** of routine tasks
  - Reduces human error in configuration and deployment
  - Enables consistent implementation at scale
  - Frees technicians for higher-value activities
- **Proactive monitoring** instead of reactive troubleshooting
  - Identifies issues before they affect users
  - Reduces system downtime and support costs
  - Enables predictive maintenance strategies
- **Centralized management** of distributed systems
  - Reduces need for physical access to all devices
  - Streamlines updates and configuration

What IT Can Learn From AV

Blurred lines

- **User experience** prioritization
  - Technical metrics don’t reflect user satisfaction
  - Simple, intuitive interfaces drive adoption
  - Technology should enhance, not hinder, collaboration
- **Real-time performance** requirements
  - Zero-tolerance for latency in critical applications
  - Quality of experience metrics beyond basic uptime
- **Environmental considerations**
  - Physical space impacts technology effectiveness
  - Human factors affect technology adoption
  - Acoustics, lighting, and ergonomics influence success
- **Creative problem-solving**
  - Custom solutions for unique requirements
  - Adaptation to varied use cases and environments

There are still barriers to convergence.

Convergence Challenges

- **Cultural Differences**
  - Career origins
  - Team structures and reporting lines
  - Decision-making process & prioritization
  - Communication styles
  - Risk tolerance levels
- **Technical Barriers**
  - Network segmentation requirements
  - Security policies and compliance
  - Bandwidth & resource competition
- **Skill Gaps**
  - IT teams lack AV expertise
  - AV teams lack IT expertise
  - Cross-training requirements
  - Knowledge transfer challenges
  - Certification differences
- **Vendor Ecosystem**
  - Proprietary vs. open standards
  - IT vendor policies vs. AV needs
  - Product lifecycle differences

Key Takeaways

Embrace Integration

- Shared infrastructure
- Unified standards
- Common workflows
- Joint planning
- Integrated monitoring & management

X-Functional Teams

- Blended skill sets
- Collaborative planning
- Joint responsibility
- Regular knowledge sharing
- Combined training

Future Focus

- Continuous education
- Outcome-oriented design
- User-centric metrics
- Balanced approaches
- Better together

Recap

What did we learn?

- What is monitoring and why should we care about it?
- Why is monitoring AV systems so hard?
- How do we design systems that are easier to monitor?
- What should we consider when selecting a monitoring platform?
- What are the steps to get started?
- Where does AV/IT convergence fit in?

Questions & Discussion

Thank you!

Fred’s LinkedIn

Presentation slides