MyRISK HyperGRC

ORACLE | Partner

Continuous Control Monitoring Implementation Series

#2 - Designing a CCM Program

White Paper

Presented by David Vohradsky
CEO and Founder

Introduction

Designing a Continuous Controls Monitoring (CCM) program is analogous to designing a building – without a solid blueprint, your structure will be shaky. Organisations can't simply buy a tool and achieve effective CCM overnight. It requires aligning the monitoring activities with business risks and compliance needs, setting clear objectives, and establishing processes. Part 2 of our series provides a comprehensive guide to planning a CCM program. We will cover how to scope which controls to monitor, how to map them to frameworks such as ISO 27001, APRA CPS 234, or Essential 8, and how to define the organisational procedures around continuous monitoring. By the end of this paper, a practitioner will have a roadmap for starting a CCM initiative that is both manageable and impactful.

1. Establishing Scope and Objectives

Every successful CCM implementation begins with clearly defined scope and objectives. This involves:

- **Understanding Obligations and Risks:** Review all relevant compliance obligations (regulations, standards, internal policies). and then extract the control areas where continuous assurance is expected or would significantly reduce your compliance burden. Simultaneously, perform or utilise a recent risk assessment – identify top cyber risks where control failure would be most damaging. The intersection of these (high-risk areas that are also compliance-relevant) often forms your initial CCM scope. Example: A bank might identify privileged user access, system configuration (hardening), third party security posture, and incident response as key areas due to regulatory scrutiny (CPS 234, CPS 230) and inherent risk.

- **Selecting Control Domains:** Decide on the "domains" or categories of controls to include. Common CCM domains include: Asset Management, Identity & Access Management, Vulnerability & Patch Management, Configuration Management, Incident Detection & Response, Endpoint/Malware Protection, Third-Party Security, etc. It's wise to start with a subset that covers both a preventive controls (e.g. access controls) and detective controls (e.g. logging and monitoring) to demonstrate value across different types of controls. Use frameworks as a checklist to ensure you're not overlooking something. ISO 27001's Annex A controls or the NIST Cybersecurity Framework functions (Identify, Protect, Detect, Respond, Recover) can serve as a reference taxonomy. Also consider any specific mandates – for instance, if you want or need to measure against the Essential 8, you may choose those eight control domains explicitly.

- **Documenting Scope:** Formally document which controls (and systems) are in scope for CCM in the initial phase. For each, write a brief description of why it's included (tie it to a risk or compliance requirement). This scoping document will help get buy-in from stakeholders by showing a rationale – e.g., "Monitoring privileged user access (ISO 27001 A.9, APRA CPS 234 requirement; high risk of data breach if misused)." It also sets boundaries, so everyone knows what is and isn't initially covered.

2. Mapping Controls to Frameworks and Requirements

For each control or domain in scope, map it to the relevant sections of your required compliance frameworks and internal policies. This mapping serves multiple purposes:

- It ensures coverage of required controls. For example, ISO 27001 might have a control for anti-malware (A.12.2.1). By mapping it, you confirm your CCM will include that ISO requirement.
- It aids in reporting. Later, you can generate reports per framework, showing continuous monitoring status for each required control in the framework.
- It highlights any controls that are important but not explicitly required by a framework (driven by risk). You might still include them, and mapping shows they are above-and-beyond tick the box compliance (good for demonstrating proactive security).

e.g. Patch Management – (ISO 27001 A.12.6.1, Essential 8 Application/OS patching, APRA CPS 234 expects timely patching.

This mapping exercise can reveal quick wins. You might discover that some requirements are already being monitored via existing systems (for example, maybe your IT team already monitors backup job success – that could be folded into CCM to satisfy a control for backups). Conversely, it will show gaps where you have requirements but no current monitoring (e.g., monitoring of supplier risk assessments might be blank – you decide if that's in initial scope or later).

3. Defining Metrics and Key Risk Indicators (KRIs)
For each control objective in scope, define how you will measure it. These become your CCM metrics or Key Risk Indicators. A well-defined metric has:

- A clear description (e.g., "Percentage of laptops with encryption enabled").
- A data source (e.g., from endpoint management tool).
- A target or threshold (e.g., ">= 98%" or "any deviation triggers alert").

Leverage any existing control tests or audit checklists – you might find pre-existing definitions. Also use guidance from best practices:

- In my ISACA paper[1] I recommend formalising control assertions and then identifying tests to verify each.
- NIST SP 800-137 (Continuous Monitoring guide) suggests metrics should be specific, measurable, assignable, realistic, and time-bound (SMART).
- If following Essential 8, the maturity model often implies metrics (for instance, "application patches applied within 48 hours" is essentially a metric).

Ensure metrics focus on effectiveness, not just presence. For example, simply counting how many vulnerability scans were run is not as useful as measuring how many high vulnerabilities remain open beyond SLA. The latter tells you if the vulnerability management control is effective.

Example Metrics:

- Access Control: "*Percentage of orphan accounts (accounts belonging to departed staff) on critical systems.*" – Source: HR vs. IT accounts reconciliation.

- Incident Response: "*Percentage of security alerts outside SLA*" – Source: SOC ticketing system, measured against a threshold (e.g., >15 minutes triggers alert to management).

- Configuration Management: "*Percentage of servers deviating from CIS baseline*" – Source: Configuration assessment tool (score below, say, 90% triggers investigation).

Document each metric, its formula, frequency of collection, and who is responsible for responding to changes in that metric. This essentially creates a monitoring playbook.

4. Selecting Technology and Tools
Choosing the right tools is critical to automate the continuous checks. The selection depends on many factors including existing infrastructure, budget, and the complexity of your environment:

- If you are heavily cloud-based, you may utilise cloud-native services: AWS Config/Azure Security Center can continuously evaluate resource configurations and send alerts for non-compliance.

- In an on-premise environment, endpoint management systems (for patch, AV status) and network monitoring tools will be key. Many organisations also deploy agents or scanners (Nessus, Qualys, etc.) for continuous vulnerability management.

- Consider unified platforms: Some GRC platforms (like MyRISK) allow you to define control tests and automatically pull data from connectors. Security orchestration platforms or SIEMs can sometimes be configured for control monitoring use-cases as well.

- Don't overlook simple solutions: even scheduled scripts with outputs to a dashboard can work, especially initially or for bespoke systems. For example, a script that checks user account lists against an "authorised list" could run daily and email differences to the security team.

When selecting tools, ensure they can scale and handle the frequency. Continuous means the tools will be running checks regularly – performance and noise considerations matter. It's wise to pilot with available tools (perhaps manually pulling data) before investing heavily in new software. This also helps develop use cases to do a formal POC using a platform.

5. Defining Governance and Processes
Governance ensures the CCM program itself is managed and improved. Key process aspects include:

- **Roles and Responsibilities:** Clearly assign who is the CCM program owner (often the CISO or GRC / cyber risk manager), who maintains the monitoring tools (maybe SecOps team), and who receives and acts on alerts (control owners in IT or business). For each control metric, there should be an "owner" accountable for that aspect of security – CCM will provide them info, but they own fixing issues. Define these in a RACI matrix (Responsible, Accountable, Consulted, Informed).

1. Vohradsky, D. (2015), "A Practical Approach to Continuous Control Monitoring", ISACA Journal, 2015(2).

- **Alert Handling Procedures:** Develop runbooks for what to do when a continuous control check fails. This might tie into incident response for serious issues. For example, if CCM finds a critical security control failure (like logging is found to be disabled on a critical system), the procedure might require an immediate incident ticket and notification to the CISO, given the risk of blind spots. Less critical alerts might be assigned for resolution within a few days. The idea is to treat control gaps with urgency proportional to the risk they pose.

- **Reporting and Review:** Decide how CCM results will be reported. Many organisations create a dashboard or monthly report showing key control metrics trends. This can feed into risk committee meetings or management reports. For compliance, you might also map results to each framework and show, for instance, "We continuously monitor 30 out of 34 ISO 27001 controls applicable – all green except 2 currently amber". Regular reviews of the CCM program itself should be scheduled. Perhaps quarterly, the team reviews if metrics need adjustment, if new controls should be added (or old ones removed if they've become low risk), and to summarise improvements (e.g., "orphan accounts metric has dropped to near-zero after we instituted CCM, indicating better control").

- **Continuous Improvement:** Embed an improvement loop. Use findings from CCM to strengthen controls. If a certain alert keeps firing, perhaps the underlying control needs fixing (not just the symptoms). Also, as new threats arise or new systems come online, update the CCM scope. The program should evolve with the organisation. The governance process should allow for change requests to add new metrics or tune existing ones, with proper approval and testing.

6. Start with a Pilot or Phase 1

A pragmatic approach is to implement CCM in phases. After design, pick a pilot domain or a limited scope to implement first. This could be a single department or system. The pilot helps validate your design assumptions. You might discover during implementation that some data is harder to get than expected, or that an alert threshold is too sensitive. These lessons can be incorporated before wider rollout. Additionally, early quick wins from a pilot (like catching a misconfiguration nobody knew about) can generate momentum and executive support to invest further in the program.

Define success criteria for Phase 1 (e.g., "demonstrate automated monitoring for at least 5 key controls and reduce manual effort by X hours/month"). Monitor and document the outcomes. Then plan subsequent phases to cover more controls or integrate more systems, using the refined design.

Case in Point: One large organisation started their CCM program focusing only on privileged account management. They mapped it to compliance (APRA and ISO requirements), set metrics (unused admin accounts, admin access change tickets vs actual changes), and pulled data from Active Directory and their ticketing system. Early on, CCM alerts identified several privilege changes that had bypassed their normal process – allowing them to quickly remove unauthorised access. This success then built confidence. Over the next year, they expanded CCM to system configurations and patch management enterprise-wide. By year two, they had reduced their audit findings by over 50%, as most issues were being caught and resolved via CCM throughout the year rather than by auditors. The design and phased approach were key to this success.

Conclusion

Designing a CCM program requires balancing ambition with practicality. By scoping based on risk and compliance, mapping controls to requirements, defining solid metrics, selecting appropriate tools, and setting up governance processes, you create a strong foundation. This upfront work pays dividends once you begin implementation, as it guides the team on exactly what to do and why. Skipping the design phase can lead to disjointed monitoring efforts that either overwhelm teams or leave critical gaps.

As you complete your CCM program design, you are ready to move into implementation. The remaining parts of this series will effectively walk through implementing CCM in specific areas, following the principles outlined here. We'll start with Asset Management in Part 3, because you can't protect or monitor what you don't know you have. Asset visibility is step zero for all of security – and a perfect place to begin continuous monitoring in practice.

By approaching CCM with a methodical plan, you set the stage for a sustainable program that delivers continuous value: ensuring your controls are effective, your auditors are satisfied, and your risk of breaches is minimised. By following this series, a cyber risk practitioner will gain the knowledge to move beyond theory and start continuously monitoring and managing controls in their environment, ultimately achieving stronger security outcomes and peace of mind with regulators.

Smarter Cyber Risk.
Automated. Empowered. Secured.