Maia Foundation : Software Development Lifecycle

Document	Detail
Title	Maia Foundation — Software Development Lifecycle
Version	1.0
Issue date	August 2026
Owner	VP Engineering
Approved by	VP Engineering; Application Security; Legal
Review cycle	Annual, or on material change to the delivery process
Classification	**External**

This document is provided as a part of information for Matillion's customers or prospects assurance activity only and will not be circulated further without Matillion's consent. For any queries, please reach out at security@matillion.com

Matillion's full security and compliance posture, including certifications, audit reports and penetration test summaries, is published at www.matillion.com/trust-center.

Contents

1. Purpose and Scope

This document describes the Software Development Lifecycle (SDLC) that governs how Maia Foundation is designed, built, tested, released and maintained. It is written for customers and prospective customers who need to understand and assess Matillion's software engineering practices as part of a supplier assurance, vendor risk or procurement review.

It covers requirements management, code standards, quality gates, testing strategy, release management, production readiness controls, engineering performance measurement, the use of AI in development, application security, software supply chain security and vulnerability management.

1.1 What Maia Foundation is

Maia is Matillion's AI Data Automation platform. It comprises three parts:

- Maia Foundation — enterprise data tools for any workload: warehouse connections, the component catalogue, the validation engine and the execution infrastructure
- Maia Team — expert AI agents working continuously as embedded agents within the platform
- Maia Context Engine — organisational data intelligence, in the form of a knowledge graph mapping the data landscape

Maia Foundation is a cloud-native SaaS platform built on a microservices architecture. It provides pipeline orchestration and transformation capabilities, connecting data warehouses, lakes and source systems through a visual development environment backed by version-controlled code artefacts. It is designed to support enterprise-scale deployments, including those handling sensitive and regulated data.

The lifecycle described here is the single engineering process operated across the Maia platform. Maia Foundation is its primary subject because that is the component customers deploy and operate against, but the same requirements management, quality gates, release controls and security assurance apply to every service Matillion builds, including those delivering Maia Team and the Context Engine.

1.2 A note on naming

Maia Foundation was previously called the Matillion Data Productivity Cloud. Published materials, certificates and audit reports issued before the change still carry the former name — including some documents currently available via the Trust Center. They refer to the same platform, the same infrastructure and the same engineering process described here.

This document describes the process as operated at the issue date shown above. Where a customer requires evidence rather than description — audit reports, certificates, penetration test summaries or a bespoke attestation — the routes to obtain it are set out in Section 15.

2. Assurance and Compliance Posture

Matillion's security and compliance posture is documented publicly at www.matillion.com/trust-center. The following third-party assurances are relevant to an assessment of the development lifecycle:

Assurance	Scope and availability
SOC 2 Type 2 SOC 1 Type 2 / SOC 3	Annual external audit against AICPA Trust Services Criteria covering security, availability and confidentiality.
ISO/IEC 27001:2022	Certified Information Security Management System. Certificate issued by a UKAS-accredited certification body and published on the Trust Center.

Assurance	Scope and availability
PCI DSS v4.0.1	Compliance documentation published via the Trust Center.
Penetration testing	Annual third-party penetration testing. Executive summaries published via the Trust Center; full reports available under NDA.
GDPR / UK DPA 2018	Compliance documented. Data processing agreements and the sub-processor list are available via the Trust Center.
HIPAA	Business Associate Agreements available for customers with healthcare data requirements.

Availability of any specific report or certificate is as published on the Trust Center at the time of request, and some items are provided under NDA.

The controls described in this document are the operational reality behind those assurances. The SOC 2 and ISO 27001 audit scopes include change management, secure development and access control, and the mechanisms described in Sections 4 to 10 are the mechanisms those audits test.

2.1 How this document relates to other Matillion documentation

This document is one part of a set published for customer assurance. It covers the software development lifecycle specifically: how software is specified, built, reviewed, tested, released and maintained, and how supply chain and vulnerability risk is managed within that lifecycle.

Where a subject is documented more fully elsewhere, this document states the control that is specific to the delivery lifecycle and points to the fuller source rather than repeating it.

For information on	Refer to
Matillion's overall security programme — security organisation, risk management, deployment architecture, identity and access management, encryption, incident response, business continuity and physical security	Matillion Security Whitepaper, Trust Center
Testing approach, tooling and quality standards in depth	Matillion Quality in Test (QiT) Strategy, Trust Center
Use of AI within the Matillion product itself	AI Security Whitepaper and AI Testing and Validation Strategy, Trust Center
Independent assurance over the controls described here	SOC 2 Type 2, ISO 27001 certificate and penetration test summaries, Trust Center
Deployment models and supported topologies	Matillion Security Whitepaper and product documentation

3. The SDLC at a Glance

Maia Foundation is developed using an agile delivery model. Engineering is organised into squads, each aligned to a product area such as Pipeline Management, Runner Services, Hub and Billing, DataOps, or AI. Squads work in iterations using Scrum or Kanban depending on the nature of their work, and each squad owns the full lifecycle of its services from design through to production operation.

This ownership model matters for quality. The people who write the code are the same people who operate it in production and carry the on-call responsibility for it. There is no separate operations team to absorb the consequences of a poorly built service, which creates a direct and continuous incentive to build services that behave well under real conditions.

Work enters the system as a ticket in Matillion's work management system, flows through three environments — Dev, Preprod and Prod — and is governed at every stage by DISPATCH, Matillion's central artefact promotion system.

3.1 DISPATCH: test-based artefact promotion

DISPATCH (Dependency Inspection System Promoting Artifacts if Test Conditions Honoured) enforces quality gates across the delivery pipeline. Every software artefact built by the CI system is immutable and versioned. An artefact can only be promoted to the next environment once all quality gates for that stage have passed.

This is test-based promotion. DISPATCH does not allow a service to advance because someone asserts it is ready; it advances because the evidence proves it is. Every promotion decision — artefact version, environment, result and triggering event — is recorded automatically and written back to the originating ticket by an automation account.

The immutability of artefacts is significant for assurance. An artefact built from a given set of commits is a fixed object. What is tested in Dev is exactly what is promoted to Preprod, and what passes UAT in Preprod is exactly what is deployed to Prod. There is no rebuild between environments, no opportunity for a build-time variable to differ, and no possibility that a manual packaging step introduces a discrepancy. The artefact that reaches production is the same binary object that was tested at every prior stage.

3.2 Release cadence

Production deployments follow a fixed weekly rhythm. User acceptance testing runs on Monday and Wednesday mornings; deployments to production follow on Tuesday and Thursday. The cadence is published in advance internally and enforced through DISPATCH.

The regularity is itself a control. A fixed rhythm creates predictable points for testing, review and sign-off, and prevents the accumulation of unreleased change that makes large, infrequent deployments inherently more risky.

4. Requirements Management

4.1 Work entry and the Definition of Ready

All work on Maia Foundation begins as a ticket. Epics capture features or significant product areas; user stories break them into units a squad can deliver within a single iteration. Bugs, chores and spike investigations use the same entry point. No development work begins without a ticket, and no ticket is picked up unless it satisfies the Definition of Ready.

The Definition of Ready is a formally maintained standard. A ticket becomes eligible to be picked up only when:

- Acceptance criteria are clear and agreed between Engineering, Product and Quality Engineering
- Test scope is agreed and recorded on the ticket, specifying which tests will be manual and which automated

- Security considerations are documented; tickets with potential threat model impact are labelled for threat modelling review, with dedicated security guild channels available for expert input
- Assumptions, dependencies and non-functional requirements are documented, covering performance, observability and analytics
- Feature flagging has been considered and scoped where required, so that incomplete work can be deployed safely behind a flag
- The subscription level is captured for any new feature
- UX designs are available for any change affecting the user interface
- The ticket has been estimated and refined; user stories are capped at eight story points, and bugs are not estimated

This standard ensures that when an engineer starts work, the requirement is unambiguous and testable. It removes the common failure mode of stories being started before they are properly understood.

4.2 Quality Engineering involvement at refinement

Quality Engineers participate in ticket refinement alongside engineers and product owners. Their role at that stage is to review acceptance criteria, identify the full test scope, and catch design defects before any code is written. A flaw found in an acceptance criterion during refinement costs nothing; the same flaw found during testing costs a sprint; found after release, it costs considerably more.

Because of this early involvement, test cases are drafted and test scope agreed before development begins. The engineer knows what the tests will check before writing a line of code, which shapes the implementation from the outset and results in fewer rewrites and fewer late-stage surprises.

Quality Engineering participation at refinement also surfaces a category of issue that engineers and product owners can miss: interaction effects between new functionality and the existing system. A Quality Engineer's perspective is oriented towards how things break rather than how they work, and applying that perspective before implementation decisions are locked in is consistently more productive than applying it afterwards.

4.3 Traceability from the first ticket

A representative Maia Foundation ticket contains a user story in standard format, an explicit in-scope and out-of-scope statement, acceptance criteria linked to the relevant design assets, and — once development is under way — the automated DISPATCH deployment trail showing the exact artefact version promoted to Dev, Preprod and Prod, each with a timestamp. This record is a by-product of the normal workflow, not something assembled for audit purposes.

5. Development and Code Standards

5.1 Branching and commit conventions

Maia Foundation is developed using trunk-based development. Feature branches are short-lived and follow a defined naming convention that links every branch directly to its ticket, providing traceability from the first commit.

Commit messages follow a semantic convention. This matters beyond readability: DISPATCH uses the commit type to drive semantic version increments automatically, so a feature commit increments the minor version and a fix commit increments the patch version. Versioning is a direct function of the code change rather than a manual step, which means the version number itself carries reliable information about the nature of a release.

5.2 Pull request quality gates

Every commit to a feature branch triggers a pull request build in the CI system. The following gates run automatically, in parallel, before a pull request becomes eligible for human review:

Gate	What it checks
Static analysis	Code quality, reliability rating, security rating and test coverage on new code
Dependency scanning	Known vulnerabilities in dependencies and licence compliance; critical vulnerabilities must be resolved before merge
Container scanning	Security scanning of container images
Unit tests	Positive, negative and boundary scenarios
Integration tests	Service dependency and input/output validation
Contract tests	Provider and consumer contract validation across service boundaries

The CI system is not advisory. A failing gate blocks the pull request; there is no self-service route around it.

5.3 Self-review and automated review

Before marking a pull request ready for review, the author is expected to conduct a self-review: checking for unnecessary complexity, verifying that all acceptance criteria are covered by tests, and confirming that the code meets team standards. This is a documented expectation rather than an informal norm. It is a lightweight but meaningful step that shifts the author's perspective from writer to reviewer and catches the class of issue that is obvious in retrospect but invisible while writing.

When a pull request is marked ready for review, an automated AI review runs against the diff, checking code quality, test coverage and adherence to team standards, and producing structured feedback before the first human reviewer looks at it. This output is explicitly advisory: it does not block a merge. It reduces noise in human review by catching straightforward issues early, allowing reviewers to concentrate on judgement calls that require contextual knowledge of the system.

The advisory status is deliberate. Automated tooling can identify patterns and flag deviations from standards, but a named human reviewer remains accountable for the merge decision. No change reaches the main branch on the strength of an automated review alone.

5.4 Use of AI in development

Matillion uses AI coding assistants as a standard part of the engineering toolchain. Their role is assistive: agents suggest and draft code, and engineers write, review and remain accountable for what is committed. Agents are contributors, not decision-makers.

Every control described in this document applies identically regardless of how a change was drafted. Code produced with agent assistance passes the same automated quality gates described in Section 5.2, requires the same two human approvals including a designated senior reviewer, and follows the same promotion, UAT and production sign-off path set out in Sections 7 to 9. There is no expedited route for AI-assisted code, no separate pipeline, and no change reaches production without a named engineer accountable for it.

The same principle governs the automated review described in Section 5.3. Automated review supplements human review; it does not replace it, and it cannot approve a merge.

AI tool usage across the engineering organisation is governed by Matillion's AI Usage Policy, which defines approved tooling, the data handling requirements that apply to its use, and the boundaries within which

assistants may be used. Tooling is subject to enterprise licensing arrangements, and the handling of code and prompt data is governed by the terms of those arrangements and by that policy.

Matillion's use of AI within the product itself is a separate matter, documented in the AI Security Whitepaper and the AI Testing and Validation Strategy, both available via the Trust Center.

6. Code Review and Approval

6.1 Approval requirements

Two approvals are required to merge any pull request into the main branch. At least one must come from a default reviewer — a named senior engineer designated for that repository. This is enforced by source-control branch protection rules and cannot be bypassed without an administrator overriding the policy, which is itself an audited action.

The default reviewer requirement means every merge into main has been assessed by someone with both the authority and the depth of knowledge to judge the change in context. Default reviewers are designated per repository, not rotated informally, which creates a consistent point of accountability.

6.2 Reviewer focus

Reviewers assess code correctness and quality, standards compliance, architectural patterns, adequacy of test coverage, and knowledge sharing. They do not manually re-test functionality already covered by automated gates — the CI system handles that. Human attention goes to what automation cannot assess: the appropriateness of a design choice, the fit of an architectural pattern, and whether test coverage is genuinely adequate rather than superficially passing.

6.3 Pull request template and hygiene

Every pull request uses a standard template requiring the author to document the reason for the change, what was tested, a link to the originating ticket, and screenshots or recordings for any change affecting the user interface. The template is not optional; pull requests without it are not accepted by reviewers. It creates a consistent record of intent and evidence for every change entering the codebase.

Pull requests without activity for approximately three months are automatically closed. Open pull requests represent in-progress work or unresolved decisions and are not permitted to accumulate indefinitely.

7. Continuous Integration and the Dev Environment

On merge to the main branch, the CI system automatically triggers the main branch build. This repeats the full pull request gate suite against the integrated codebase, verifying that newly merged code behaves correctly alongside everything else merged since the last build. On success, the artefact is deployed to the Dev environment.

Gate at Dev	Purpose
Static analysis	Reliability and security ratings, coverage on new code
Dependency scanning	Known vulnerabilities and licence compliance
Unit tests	Positive, negative and boundary scenarios
Integration tests	Service dependency and input/output validation
Component tests	Behaviour of a single microservice in isolation
Container scanning	Security scanning of built container images

All gates must pass before the artefact is registered in DISPATCH as eligible for promotion to Preprod. If any gate fails, the artefact is not registered and cannot progress. There is no manual override path for a failing artefact at this stage.

Once deployed, DISPATCH notifies the owning team and updates the originating ticket with the exact artefact version and deployment result — recording the version deployed, the environment, the outcome and the previous version. No engineer writes that record; DISPATCH writes it. The central component test suite then runs against Dev on a schedule, every two hours from 07:00, covering Snowflake, Redshift and Databricks across the supported runner types.

8. Preprod Promotion and User Acceptance Testing

8.1 Controlled promotion

Promotion from Dev to Preprod is not automatic. An engineer or release manager triggers it explicitly through DISPATCH. At the point of triggering, the approver must confirm whether the deployment requires production downtime and whether it introduces schema changes. That information is used by Release Engineering and SRE to prepare for the downstream production deployment.

Before promoting, DISPATCH performs dependency health checks, verifying that all known dependent services are present in the artefact store and running healthily in Preprod. If a dependency is absent or unhealthy, the promotion is blocked. This prevents a well-known failure mode in which a service is deployed into a broken environment because a dependency was not ready.

8.2 Quality gates at Preprod

Gate	Purpose
End-to-end tests	Full service E2E suite executed post-deployment
Component regression pack	Regression suite running on a recurring schedule
Smoke tests	Critical path validation
Contract compatibility	Confirms the artefact is contract-compatible with all other services in the environment
Central QE monitoring	The central Quality Engineering team monitors failures and follows up directly with service owners

Component tests run in Preprod every four hours from 08:00, with selected suites running hourly. Performance tests run daily against Preprod overnight.

8.3 UAT entry criteria

Before a service can enter UAT it must satisfy the following conditions. These are checked and confirmed by the team, not assumed:

- The test plan has been updated with all significant changes by 09:45 UK time on the UAT day
- The team has signed off the service as ready for UAT
- All quality gates are passing — unit, integration, component, contract, end-to-end and smoke — as asserted via DISPATCH
- Feature flags are in a known working state

- The development team has completed exploratory testing in Dev and signed off the ticket

A service that has not met these conditions does not enter UAT. It remains in Dev or Preprod and is carried to the next release cycle. This is a hard gate, not a matter of negotiation on the day.

8.4 The UAT cycle

UAT runs on Monday and Wednesday mornings to a fixed schedule:

Time	Activity
09:00	Release Engineering imposes a Preprod deployment freeze via DISPATCH. No further deployments to Preprod are permitted until the freeze is lifted.
10:00 – 12:00	Central Quality Engineering executes UAT against the frozen Preprod environment.
12:00	The Release Acceleration Group meets. Engineering Managers and Product Owners attend for any service with changes in Preprod. Attendance is mandatory where a service has failed UAT.
12:00	Release Engineering makes the GO or NO-GO decision on production promotion based on the UAT results.
On completion	The UAT ticket transitions to Done, automatically triggering the SRE production deployment ticket for the following morning.

The mandatory attendance rule matters. It ensures failures are not quietly deferred: they are discussed, owned and resolved in front of the people accountable for them before the GO/NO-GO decision is made.

9. Production Deployment and Change Management

9.1 Deployment schedule and sign-off

Production deployments run on Tuesday and Thursday mornings. Approved artefacts are deployed between 09:00 and 11:00, and production smoke test sign-off is executed between 11:00 and 12:00.

Gate at Prod	Purpose
Release / UAT testing	Automated tests running hourly; manual UAT on Monday and Wednesday prior to deployment
Contract testing	Contract compatibility verified across all service dependencies
Version compatibility	Confirms all versions already in production are compatible with the new artefact
Smoke tests	Post-deployment critical path validation
Synthetic tests	Automated user journeys running hourly
ETL pipeline tests	End-to-end pipelines running hourly in Preprod pre-deployment
Owner sign-off	The service owner and all dependency owners must approve before deployment proceeds

The owner sign-off step warrants description. Before any service is deployed to production, DISPATCH collects explicit approval from the service owner and from the owners of every service it depends on. This ensures the people closest to each service have the opportunity to flag concerns before deployment. It is not a rubber stamp:

an owner with concerns raises them in the same thread, and the deployment does not proceed until they are addressed.

Component tests continue to run in production every four hours from 08:30, covering the supported data platforms.

9.2 Change request process

Any non-routine change to customer-facing production infrastructure requires a formal change request raised through the internal service desk. Changes are risk-tiered:

Risk tier	Handling
High	Requires a scheduled maintenance window, communicated in advance via the public status page.
Medium	Applied outside EU and US business hours, with an agreed runbook and rollback procedure, notified to engineering channels three days and one day in advance.
Low	Scheduled by SRE at their discretion within normal deployment patterns.

All production changes apply a four-eyes principle: the change is applied by one engineer and peer reviewed by another. Pull requests that delete production resources require sign-off from two named SRE leaders. SRE verifies that every production change has been successfully applied in a lower environment first. Production is never the first place a change is tried.

9.3 Hotfix process

A hotfix follows a separate, documented DISPATCH flow. A build-only pipeline runs from the hotfix branch, producing a versioned artefact, and the build is verified before a deployment is committed. The deployment can be triggered by a member of the incident channel, and DISPATCH checks authorisation before proceeding. The hotfix artefact passes through a compressed but documented gate set, and the promotion is recorded automatically against the incident ticket in exactly the same way as any other deployment.

The significant point for assurance is that an emergency change does not escape the audit trail. The gate set is compressed under incident conditions; the record is not.

10. Release Records and Traceability

10.1 Test plan structure

Each release has an associated release test plan ticket. This ticket is the definitive record of what was included in a given release and what was tested. It records the Maia Foundation version being released, every service artefact in the release with its exact version in Dev, Preprod and Prod at the time of release, and every feature ticket included in the release with its own test cases.

The standard is five test cases per user story: happy path, validation, error handling, edge cases and regression. These are documented directly on the test plan ticket rather than in a separate system, so the record and the work are never out of step.

A typical release test plan covers more than thirty services across the product estate, each with its current and target versions across all three environments, and links the CI pipeline that executed the tests. A single such record provides an auditor with the complete picture for a given release: what changed, which artefact version carried the change, what environment versions were aligned at the time of release, and what was tested and how.

10.2 The full traceability chain

The combination of the release test plan and the automated DISPATCH records on individual feature tickets creates a complete, auditable chain:

Requirement record (acceptance criteria, test scope, Definition of Ready sign-off) → development (delivery plan, feature branch, commits) → pull request approval (two reviewers, CI gates passing) → CI pipeline (main branch build, full gate suite) → Dev (artefact registered in DISPATCH, component tests scheduled) → Preprod (dependency health check, end-to-end and smoke tests, UAT) → Prod (owner sign-off, smoke test sign-off, post-deployment monitoring).

At every step the artefact version is recorded alongside the environment, the result and the timestamp. Nothing in this chain requires manual record-keeping: it is generated automatically by DISPATCH and the CI system as a by-product of normal operation, which means the evidence is complete whether or not anyone anticipated being audited.

Because semantic version increments are driven by commit type, it is possible to infer from a version number alone whether a release contains only fixes or also introduces new functionality. Combined with the release test plan, which records the specific artefacts and their versions across all three environments at the point of release, this provides a clear and auditable picture of what changed between any two production releases.

11. Production Readiness Review

11.1 Purpose

Before any new Maia Foundation service is released to private preview, and again before general availability, it must pass a Production Readiness Review (PRR). The PRR is the mechanism by which SRE formally accepts operational responsibility for a service. No service reaches production without it.

11.2 Process

Epics are created from the PRR template at the start of the development lifecycle, alongside non-functional requirements — not at the end. One epic is assigned to the development team and one to SRE. Both teams work through the checklist, marking each item done when the standard is verified.

Before release, SRE and the development team jointly conduct a walkthrough session of at least nine hours: approximately four and a half hours on the application and four and a half on cloud infrastructure. This is not a presentation. It is a working review in which SRE engineers ask questions, probe assumptions and test runbooks against realistic failure scenarios.

Completion is tracked on a PRR dashboard giving a real-time view of readiness across all services. A partially ready service is visible as such: its incomplete items are listed, owned and tracked until resolved.

11.3 The sixteen standards

Standard	What it verifies
SLAs and SLOs	Verifiable service level agreements and objectives defined and implemented
Internal documentation	All operational documentation accessible to the SRE team
Deployment architecture	Architecture reviewed against SaaS standards and scalability requirements
Cost optimisation	Infrastructure cost efficiency validated against expected usage patterns
Incident management	On-call processes, runbooks and escalation paths in place and tested

Standard	What it verifies
Reliability	Fault tolerance and scalability verified under expected and peak load
IaC and configuration	Infrastructure-as-code for all resources; no manual configuration in production
Observability	Internal state inference capability, structured logging, monitoring platform integration
Logging	Structured logs, no sensitive data logged, integrated into the observability platform
Monitoring	Proactive and actionable alerting configured for known failure modes
Continuous integration	Code versioned, builds automated, artefacts versioned and labelled
Quality control	Artefacts meet quality gates and Quality in Test standards
Continuous delivery	Deployment automated, immutable and zero-downtime capable
Database management	Schema lifecycle managed, least-privilege access, audit logging in place
Security	Security scanning in CI, access controls verified, vulnerability management active
Disaster recovery	Recovery procedures tested and documented with validated recovery time objectives

The PRR is not a one-time gate applied at the end of a project. Creating the epics at the start of the lifecycle means production readiness items — observability instrumentation, runbook preparation, disaster recovery procedures — are tracked as deliverables alongside feature work rather than retrofitted before launch.

12. Engineering Performance Measurement

Matillion measures engineering performance using the DORA (DevOps Research and Assessment) framework, reported at service and team level and rolled up to an engineering-wide view.

Metric	Definition	Elite threshold
Deployment frequency	How often code successfully reaches production	Multiple times per day
Lead time for changes	From first commit to production deployment	Under 1 day
Change failure rate	Proportion of deployments causing a production incident	Under 5%
Failed deployment recovery time	Time from a production failure to full recovery	Under 1 hour

The thresholds shown are the published industry benchmarks for elite performance against which Matillion measures itself; they are the measurement standard rather than a statement of performance in any given period.

The following flow metrics are tracked alongside DORA at team level:

Metric	Elite threshold
Pull request cycle time	Under 24 hours
Time to first review	Under 4 hours
Batch size	Under 200 lines of code per pull request

Metric	Elite threshold
Defect escape rate	Under 10%
Pipeline failure rate	Under 5%

Batch size deserves specific mention. The threshold of under 200 lines of code per pull request reflects a deliberate quality philosophy: smaller changes are easier to review, easier to test and easier to revert. Large pull requests are a known risk factor, both because they are harder to review thoroughly and because they tend to bundle multiple concerns, making it difficult to isolate a cause when a problem emerges. Tracking batch size as a quality metric sustains the delivery discipline that makes a twice-weekly release cadence safe.

Defect escape rate — the proportion of defects reaching production rather than being caught in testing — is a direct measure of testing effectiveness. Tracking it over time at service and team level allows engineering leadership to identify patterns: a rising escape rate for a particular service may indicate gaps in coverage or a need for additional Quality Engineering capacity.

These metrics serve two purposes in a supplier assurance context. They demonstrate that Matillion's engineering organisation operates at a measurable level of delivery performance, and they act as leading indicators of quality risk. A rising change failure rate or a lengthening lead time signals that a team or service is under stress, and does so before that stress becomes a production incident.

13. Application Security

13.1 Security standards

The OWASP Application Security Verification Standard (ASVS) 4.0 is the baseline for functional and non-functional security requirements across Maia Foundation. Every new feature is assessed against ASVS-defined requirements before it is built, not after. Matillion's application of the standard establishes which verification level applies to which categories of functionality, providing a structured and auditable basis for security requirements rather than relying on ad hoc review.

13.2 Security gates in the pipeline

Static analysis, dependency scanning and container scanning run as gates in every CI pipeline; the scanning programme itself is described in the Security Whitepaper. What matters for the lifecycle is that these are blocking gates rather than reports. Critical vulnerabilities identified in dependencies must be resolved before a pull request can be merged. Where a critical finding cannot be remediated immediately — for example where no upstream fix yet exists — it may proceed only under the formal waiver process set out in Section 14.4, which requires approval from VP Engineering and the Application Security Lead, carries an expiry date and a monitoring commitment, and is recorded. There is no informal route past a critical finding. High-severity findings are tracked and remediated within the SLAs in Section 14.

Dependency scanning also covers licence compliance. Dependencies carrying licences incompatible with Matillion's obligations are flagged for legal review before they can enter the codebase.

13.3 Threat modelling

Matillion's threat modelling programme is described in the Security Whitepaper. The lifecycle-specific point is where it sits in the workflow: threat modelling is a formal step in the Definition of Ready, and tickets with potential threat model impact are labelled during refinement so they are flagged for security review before work begins. It is not a separate review bolted on at the end, which is what makes it reliable — a security review that sits outside the delivery workflow is one that gets skipped under schedule pressure.

13.4 Penetration testing

Matillion commissions routine third-party penetration testing and operates a bug bounty programme; both are described in the Security Whitepaper. Executive summaries are published via the Trust Center and full reports are available under NDA. Findings from either route enter the same vulnerability management process described in Section 14, with the same ownership, remediation SLAs and escalation path as any other finding.

14. Software Supply Chain Security and Vulnerability Management

14.1 Software Bill of Materials

Software Bill of Materials (SBOM) generation is integrated into the CI process for Maia Foundation services. SBOMs are produced in the CycloneDX standard format, covering both container images and application dependencies, and are uploaded automatically to a dedicated supply chain security platform as part of the build. The approach gives visibility of what is actually shipped to preprod and production, rather than a periodic snapshot of what was intended to ship, and coverage is maintained across the service estate as it evolves.

SBOMs are also published alongside product release notes, so that customers can inspect the component inventory of a given release directly.

14.2 Vulnerability detection and ownership

When a new CVE is published that matches a component in a service's SBOM, it is detected on the next scheduled scan and an alert is raised directly to the owning engineering team, including the CVE identifier and CVSS score, the affected component and version, the fixed version where one exists, and the remediation deadline.

The operating principle is that engineering teams own the vulnerabilities in their services. The Application Security function owns the policy, tooling and escalation path; it does not remediate on behalf of engineering teams. Teams are expected to triage new findings within two business days and to review their vulnerability posture before each sprint, treating a clean posture as a quality metric alongside test coverage and build stability.

14.3 Remediation SLAs

Remediation deadlines follow a risk-based model informed by NIST SP 800-40 Rev. 4 and prioritise entries in the CISA Known Exploited Vulnerabilities (KEV) catalogue in line with CISA guidance:

Severity	CVSS score	Remediation SLA
Known Exploited (CISA KEV)	Any	7 days
Critical	9.0 – 10.0	15 days
High	7.0 – 8.9	30 days
Medium	4.0 – 6.9	90 days
Low	0.1 – 3.9	180 days

Presence in the CISA KEV catalogue overrides CVSS score: a finding listed there carries the seven-day SLA regardless of its score, and is treated as a security incident. The SLA clock starts when the vulnerability is first detected, not when the alert is acknowledged.

CVSS score alone does not determine priority. Teams also assess exploitability (including EPSS score and KEV listing), reachability of the vulnerable code path, and customer exposure of the affected component.

14.4 Waivers and escalation

Where a finding will not be remediated within its SLA, a formal waiver must be raised on the ticket. Every waiver carries a justification, supporting evidence, a proposed expiry date for time-limited acceptances, and a monitoring commitment for the interim.

Waivers are categorised by the VEX status being asserted, and the signer follows the status rather than the CVSS score. This reflects the substance of the decision: a claim that a CVE does not apply, or is already patched despite the version string, is a technical evidence question, while accepting a real risk is a leadership decision.

VEX status	When it applies	Signer authority
not_affected	The CVE does not apply to this deployment — unreachable code path, wrong operating system or architecture, SBOM mislabel, or test-scope only	Application Security
affected	The service is genuinely vulnerable and the finding will not be remediated in the short term, whether time-limited (no upstream fix yet) or permanent (component due for retirement, remediation cost disproportionate to risk)	Engineering leadership, escalated by severity — see below
fixed	The service is on a patched version. Where the SBOM reflects the patched version the scanner auto-resolves the finding and no sign-off is required; where a distribution has backported the fix without changing the upstream version string, the claim is recorded on the ticket and countersigned.	Application Security (for distribution-backport claims)

For affected waivers, the leadership tier scales with the severity of the finding being accepted:

Severity of finding	Approval authority for an affected waiver
Known Exploited (CISA KEV)	Director of Engineering
Critical	Director of Engineering
High	Engineering Manager

Application Security runs a weekly waiver review. In this cycle, findings that require a waiver are identified, triaged into not_affected, affected and fixed streams, and progressed to closure. Not_affected claims and distribution-backport fixed claims are signed off in the review by Application Security. Affected waivers are packaged and chased with the engineering owner and the leadership signer identified above until a decision is recorded. Waivers approaching or reaching their expiry date are re-raised for a fresh decision in the same cycle.

Time-limited waivers carry an expiry date. When it is reached, the finding is re-raised and a new waiver decision is required — an accepted risk does not quietly become a permanent one.

Where an SLA deadline passes with no resolution and no approved waiver, a defined escalation path runs from the team lead, to the Engineering Manager, to Director of Engineering, and the breach is recorded in the monthly Application Security report.

14.5 Reporting

The Application Security function produces a monthly vulnerability posture summary per service, a monthly SLA compliance report, and a quarterly aggregate dashboard for security leadership. Customer attestation reports covering vulnerability posture are available to enterprise customers on request, and the same data source supports SOC 2 and ISO 27001 audit cycles.

15. Customer Change Control and Evidence

15.1 Runner updates and customer change control

Maia Foundation is offered in a range of deployment models; those models, and the architecture of each, are documented in the Security Whitepaper and the product documentation. This section covers only the part that belongs to the delivery lifecycle: where a runner executes inside the customer's own environment, how updates to that runner are controlled and what change-control evidence the customer receives.

Runner updates follow a pull model. Matillion publishes each new runner version and the customer chooses when to apply it. Updates are not forced. Release notes for each feature are published via the public documentation changelog, together with the minimum runner version required to support it, so customer engineering and assurance teams can review what has changed before deciding whether to apply an update.

Matillion maintains backward compatibility policies for runner versions: a given control plane version continues to operate with a defined range of runner versions, so a control plane release does not force a runner update. Supported version ranges and release tracks are published in the product documentation, which is the authoritative source for the policy in force at any time.

This creates a clear division of change control. A runner update is a planned, change-controlled event inside the customer's own change management process, supported by Matillion's release notes and version-based artefact tracking. Control plane updates are governed by Matillion's change management process, with the DISPATCH audit trail and release test plans providing the evidence record, and the public status page and changelog providing advance notification of significant changes.

Every runner version a customer can apply has passed the full lifecycle described in this document before it is published: the same quality gates, the same UAT cycle, the same production sign-off and the same supply chain and vulnerability controls set out in Section 14. The customer controls the timing; the assurance over what is being applied is the subject of this document.