



Secure Development Environment Guidelines

Control Page

ATTRIBUTE	DETAIL
Document Title	Secure Development Environment Guidelines
Document ID	OG-SDLC-001
Version	1.1
Effective Date	November 2, 2025
Next Review Date	November 2, 2026 (Annual)
Policy Owner	Chief Security & Compliance Officer (CSCO)
Approved By	Chief Executive Officer (CEO)
Classification	Confidential – Internal Use Only
Applicability	Engineering, DevOps, IT, Security, and all personnel involved in software development and deployment



Compliance Frameworks	PCI DSS v4.0, ISO/IEC 27001:2022, NIST SP 800-218 (SSDF), OWASP ASVS v4.0
Distribution List	Engineering, DevOps, IT, Security, Product Management

Approvals

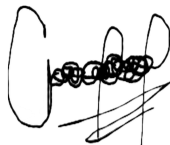
NAME	POSITION	SIGNATURE	DATE
Oleksii Bondarenko	Chief Security & Compliance Officer		1 November 2025
Charles Odhiambo	Technical Lead		1 November 2025
Kevin Kegera	Chief Commercial Officer		1 November 2025
Preston Obadan	Chief Executive Officer		1 November 2025

Table of Contents

1. Introduction	3
2. Secure Development Environment	3
2.1 Components of a Secure Development Environment	3
2.2 People	4
2.2.1 Employee Screening	4
2.2.2 Development Competence and Training	4
2.2.3 Segregation of Duties	5
2.3 Process	5
2.3.1 Separation of Development, Testing and Operational Environments	5
2.3.2 Access Control	6
2.3.3 Development Procedures	7
2.3.4 Version Control	7
2.3.5 Code Monitoring	8
2.3.6 Backups	8
2.4 Technology	9



2.4.1 Programming Languages and Coding Guidelines	9
2.4.2 Secure Repositories	10

1. Introduction

This document establishes comprehensive guidelines for the establishment, operation, and maintenance of a Secure Development Environment (SDE) for Orchestrate Global's payment orchestration platform. A secure development environment is foundational to producing software that is resilient to threats, compliant with regulatory mandates (notably PCI DSS v4.0.1), and aligned with industry best practices.

The software development lifecycle (SDLC) encompasses the following core stages:

- Business Requirements Specification
- System Design
- Development
- Testing

Each stage must be supported by a development environment that enforces confidentiality, integrity, and availability of code, configurations, and data—particularly when handling cardholder data (CHD). This guideline ensures that people, processes, and technologies are aligned to mitigate risks across the SDLC.

2. Secure Development Environment

2.1 Components of a Secure Development Environment

A robust SDE integrates controls across three interdependent domains: People, Process, and Technology. The required security posture is determined through a formal risk assessment that considers:

- Sensitivity and classification of data (e.g., CHD, PII)



- Criticality of the application to business operations
- Applicable legal, regulatory, and contractual obligations (e.g., PCI DSS, GDPR)
- Integration points with existing systems (e.g., processors, banking APIs)

All controls implemented must be traceable to specific risks and documented in project security plans.

2.2 People

Human factors are critical to maintaining environment integrity and code security.

2.2.1 Employee Screening

All personnel with access to development environments—especially those handling payment logic—must undergo rigorous pre-employment screening, including:

- Professional and personal reference verification
- Academic and certification validation
- Employment history confirmation
- Criminal background checks (aligned with local employment law)
- Credit checks for roles involving financial systems

For ongoing roles involving CHD, refresh criminal and financial checks every two years or prior to assignment to high-risk projects. Legal counsel must approve all screening protocols.

2.2.2 Development Competence and Training

Developers must demonstrate proficiency not only in programming languages (e.g., Elixir, Node.js) but also in secure coding practices. Competency is validated through:

- Quarterly secure coding assessments (aligned with OWASP Top 10 and PCI DSS Requirement 6)
- Mandatory certification in payment security fundamentals
- Hands-on labs covering CHD protection, input validation, and cryptographic implementation



Training records are maintained by HR and reviewed during internal audits.

2.2.3 Segregation of Duties

To prevent fraud, error, and unauthorized changes, duties are strictly separated:

FUNCTION	ASSIGNED ROLE
Code Development	Developer
Code Review	Peer Developer + Security Analyst
Test Execution	QA Engineer
Deployment Approval	DevOps Engineer + Operations Lead
Production Access	Operations Team Only

No individual may perform development, testing, and deployment functions for the same component.

2.3 Process

Structured processes ensure consistency, auditability, and security enforcement.

2.3.1 Separation of Development, Testing and Operational Environments

Orchestrate Global maintains three distinct environments, each with strict data and access controls:

ENVIRONMENT	PURPOSE	DATA TYPES	SECURITY CONTROLS
-------------	---------	------------	-------------------



Development	Active coding	Synthetic or anonymized only	Network isolation, developer-only access
Staging	Pre-production validation	Anonymized production-like data	Production-mirrored security, limited access
Production	Live payment processing	Real cardholder data (CHD)	Full PCI DSS controls, least-privilege access

Data movement between environments is prohibited unless anonymized via approved tokenization or masking tools. Environment cloning requires Security Team approval.

2.3.2 Access Control

- Developers use dedicated, non-privileged accounts for development work.
- No developer accounts have access to production environments.
- Access is granted via role-based access control (RBAC) and reviewed quarterly.
- All access is logged and monitored via SIEM; anomalies trigger alerts.
- MFA is enforced for all remote access (e.g., VPN, Git repositories).

2.3.3 Development Procedures

All development activities follow documented, standardized procedures covering:

- Branching and merging strategies (GitFlow)
- Code review workflows
- Security gate checkpoints (per OG-SEC-006)
- Incident reporting for suspected vulnerabilities

Deviations require CTO and CISO approval and are logged as exceptions.

2.3.4 Version Control



A centralized, secure version control system (e.g., GitHub Enterprise with SSO and branch protection) is mandatory from project inception. Key requirements:

- All source code, IaC, and configuration files are stored in version control.
- Automated versioning: Semantic versioning (SemVer) enforced via CI pipeline.
- Immutable history: Rewriting history (e.g., `git push --force`) is disabled.
- Signed commits: All commits require GPG or S/MIME signing.
- Branch protection: Main and release branches require PRs, approvals, and status checks.

2.3.5 Code Monitoring

Continuous monitoring detects unauthorized or anomalous activity:

- File integrity monitoring (FIM) on repositories and build artifacts
- Real-time alerts for mass deletions, unusual commit patterns, or access from new geolocations
- Integration with SIEM for correlation with user behavior analytics (UEBA)
- Weekly audit logs reviewed by Security Team

2.3.6 Backups

Development artifacts represent significant intellectual and operational value.

Backup strategy:

- Daily incremental backups of all repositories, CI/CD configurations, and environment definitions
- Weekly full backups retained for 90 days
- Backups encrypted at rest (AES-256) and in transit (TLS 1.3)
- Restoration tested quarterly as part of DR drills
- Backup access restricted to IT Operations and Security

2.4 Technology

Technical controls enforce secure development at the tooling and infrastructure layer.

2.4.1 Programming Languages and Coding Guidelines



Only approved languages and frameworks are permitted:

CATEGORY	APPROVED OPTIONS	SECURITY REQUIREMENTS
Backend	Elixir, Node.js	Latest LTS versions; security patches within 7 days
Databases	PostgreSQL, MySQL	TLS encryption, row-level security, audit logging
Cryptography	AWS KMS, HSM, OpenSSL	FIPS 140-2 compliance; no custom crypto
Authentication	OAuth 2.0, JWT	MFA support; token expiration $\leq$ 15 minutes

Secure coding guidelines—derived from PCI DSS, OWASP ASVS, and internal standards—are embedded in IDEs via linters and enforced in CI pipelines.

2.4.2 Secure Repositories

All code and configuration reside in secure, access-controlled repositories with:

- Least-privilege access: Developers only access repos for their assigned projects
- Repository structure: Standardized layout (e.g., `/src`, `/tests`, `/infra`, `/docs`)
- Secrets management: No credentials in code; secrets injected via HashiCorp Vault or AWS Secrets Manager
- Automated scanning: SAST, SCA, and license compliance on every push
- Audit trail: Full history of commits, PRs, approvals, and deployments retained for 3 years

Repository creation requires approval from the Development Manager and Security Architect.



ORCHESTRATE

This document is the intellectual property of Orchestrate Global. Unauthorized reproduction or distribution is strictly prohibited.